Escola Superior d'Enginyeries Industrial, Aeroespacial i Audiovisual de Terrassa

Projecte Final de Grau

Títol: Programació de Raspberry Pi per al control per àudio d'un entorn d'efectes visuals.

Alumne: Víctor Delgado Martín

Titulació: Grau en Enginyeria de Sistemas Audiovisuals.

Tutor : Ignasi Esquerra Llucia

Índex:

1. INTRODUCCIÓ

Quant entrem a veure un espectacle a una sala de concert o discoteca esperem principalment escoltar un bon producte musical, però hi ha un munt de factors secundaris que faran d'aquesta una experiència inoblidable.

Un d'aquets factors és sens dubte l'experiència visual de l'espectador. Fa molts anys que existeix una industria molt avançada de productes com ara els caps mòbils, els *rockets* de llum al·lògena , flash, etc. Però des de fa relativament poc temps existeix una tecnologia a base de leds que està revolucionant el sector de les discoteques.

Les matrius o pantalles de led permeten als dissenyadors d'espectacles tenir un producte que sigui desmuntable, econòmic de mantenir i alhora amb un gran ventall de possibilitats d'efectes. Els espectadors per a la seva banda gaudeixen d'un espectacle diferent a cada sessió degut a la naturalesa a base de blocs que té aquesta tecnologia i també tenir una bona visió del efecte des de qualsevol punt del recinte gracies al disseny del led.

Fa no molts anys quan es dissenyava una sala, el propietari tenia que pagar uns aparells molt costosos i rudimentaris que tenia que tenir un tècnic específic a cada espectacle, i que conegués i sàpigues controlar aquests llums. Més tard va aparèixer el protocol DMX que va simplificar molt el control d'uns efectes visuals als directors de sala. Aquest protocol permet automatitzar diferents tipus de productes dins d'una sala mitjançant un sol cable de 512 canals. Això sumat a

les taules de llums, permetien programar amb un llenguatge de control de *dimmers*, una taula amb tots els programes que necessitaria cada funció. Un cop programat nomes caldria una persona que executes el programa específic.

Quant va aparèixer la tecnologia led el primer pas va ser canviar els aparells de la xarxa DMX. Així doncs els "*flash*" van passar de bombeta incandescent a bombeta led, els *rockets* van passar de bombeta al·lògena a bombeta de leds, i així amb tots els efectes. Això provocava tres coses molt importants, els productes baixaven de preu, aquets tenien un temps útil més llarg i finalment baixava el consum elèctric de la sala.

Una mica més tard es va començar a treballar la idea de multiplexar un número concret de leds per produir mòduls de leds. Aquest punt va ser molt important ja que es podien dissenyar pantalles tan grans com es desitges. Aquesta tecnologia també té la capacitat de muntar blocs distribuïts, com per exemple un escenari en el que es poden posar en diferents alçades i posicions per fer efectes 3D. En espais més reduïts la gran avantatge és que el led es veu bé, amb tota la seva lluentor des de qualsevol angle.

Per una altre banda la tecnologia de control d'efectes visuals al ritme de la música és relativament nova. Alguns aparells DMX led o controladores d'il·luminació tenen un sensor que localitza els impactes del so i pot realitzar un efecte de *strobo* al moment que rep un cert nivell. El control d'un senyal d'àudio processat en temps real és ja un efecte a l'abast de grans productores de festivals i concerts.

La meva experiència professional de més de 10 anys en sales de discoteques em van fer adonar del poc interès dels propietaris en vers als efectes visuals. El problema més gran que es troben és que adquireixen un material car que després no saben aprofitar per culpa d'unes programacions rudimentàries o per la falta de tècnics que verdaderament estiguin in situ adaptant els efectes a cada

sessió. La meva idea de projecte és facilitar aquesta feina proporcionant un sistema visual guiat pel ritme de la música.

Gràcies als meus estudis cursats a l'ESCOLA de Terrassa, he adquirit els coneixements base per començar a investigar com funciona un sistema així.

1.1 Disseny:

L'estructura del projecte es divideix en dos fases, primerament trobem la implementació d'un sistema hardware i seguidament la codificació del software que executarà les accions.

1.1.1 Hardware

- La implantació i disseny de la connexió entre l'ordinador i la Raspberry mitjançant dues plaques d'Arduino com a controladors de la comunicació entre equips. Aquesta transmissió y recepció anirà muntada sobre un bus de comunicació I2C.
- La implementació de la comunicació entre la Raspberry i els perifèrics d'efectes visuals, ja sigui un monitor LCD amb entrada HDMI o un grup de mòduls led a forma de pantalla. Aquest últim tindrà que tenir un estudi de corrent perquè independentment de la quantitat de mòduls que hi hagi li arribi suficient corrent.
- Creació d'una caixa estàndard per el prototip on dipositar tots els circuits i connexions d'una manera segura i que només tingui entrades i sortides a nivell comercial.
- La estructura del hardware quedaria com la figura 1:

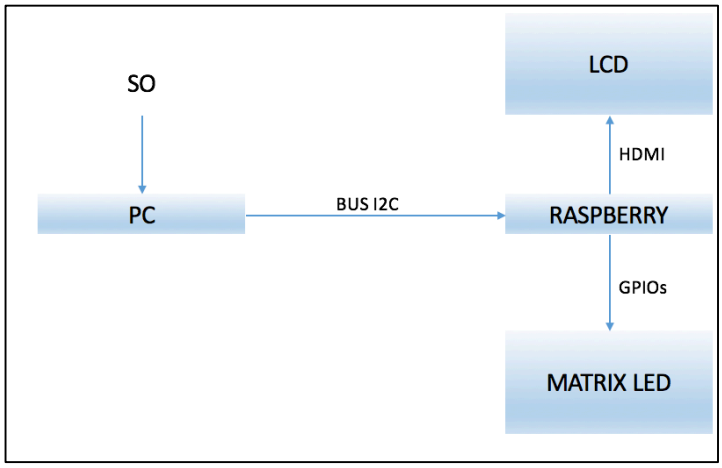


Figura 1. Esquema hardware

1.1.2 Software

- Disseny i estructura d'un codi de IDE Processing que obtingui un senyal d'àudio extern, la processi i l'enviï. Programar les funcions necessàries per que entre l'aplicació i l'usuari, que sense coneixements previs del projecte o de programació pugui fer servir el programa. Per tal de crear una aplicació visualment agradable per l'usuari, Processing també executarà un programa on mostrarà el número de programes, els colors que disposa ...
- Disseny i implementació del codi dels dos Arduino . El microcontrolador de transmissió(TX) haurà de tenir un codi que processi la informació que arriba des de Processing mitjançant el port serial, escrigui en un array de comunicació, tota aquest missatge i l'enviï per el bus I2C mantenint l'estructura de les dades. El segon microcontrolador encarregat de la recepció(RX) no haurà de processar cap missatge, simplement la seva finalitat serà anar rebent del bus I2C i anar enviant automàticament per el port serial de la Raspberry.
- EL disseny i la programació d'un script basat en Python que per una banda rebi la informació mitjançant el port serial i la processi per poder treballar amb ella. I per altre banda un conjunt de funcions que a partir d'aquest senyal rebut des de Processing amb àudio processat i informació de

l'aplicació mostri pels perifèrics diferents programes d'il·luminació. La estructura quedaria com la següent figura(2):

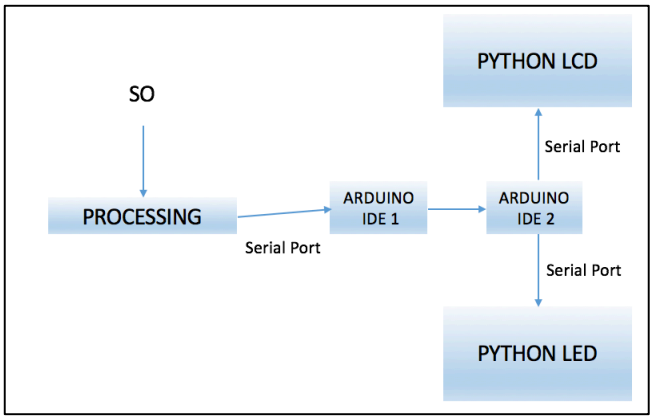


Figura 2. Esquema Software

1.2 La motivació

La motivació d'aquest projecte ve generada per la constant evolució y creixent demanda de sistemes d'il·luminació interactius.

Actualment podem veure per tot arreu com aquests productes ens acompanyen dia a dia: als cartells de publicitat, als platós de televisió però sobretot s'han tornat indispensables per els espectacles musicals com els festivals de música electrònica.

Existeix una evolució cada dia més gran en aquest sector i la gran culpa és gràcies a l'aparició del led al mercat i la constant evolució de microprocessadors, cada cop més petits, eficients i econòmics.

Els leds estan substituint a tot tipus de bombetes i equips d'il·luminació, però concretament les matrius LED és el que em crida més l'atenció per totes les aplicacions interesants que es poden aplicar i les seves característiques econòmiques. Generar per tant un sistema d'interacció entre l'usuari i una matriu led controlada per àudio processat en temps real és un projecte molt important

perquè aquest tipus de mètodes es puguin assentar al mercat com una bona opció per a qualsevol empresari del món dels espectacles.

2. ESTAT DE L'ART

La principal tecnologia que es pot assimilar al d' aquest projecte és Adafruit industries.

Aquesta empresa, originaria de USA es caracteritza per fabricar components electrònics de gran qualitat i de lliure programació. També disposa de exemples que proposen gent que compra aquest hardware.

Això té un inconvenient i és que els productes no estan dissenyats per comercialitzar ,sinó per una electrònica de projectes de casa. En aquest projecte s'arreglarà aquest handicap ja que es dissenyarà un prototip que es pugui comercialitzar.

Per un altre banda hi ha diversos distribuïdors de tecnologia led com per exemple l' empresa de Terrassa Led Dream. Aquesta és una empresa especialitzada en comercialitzar pantalles de led per diferents propostes; Escaparates de tendes comercials, Informatives per gimnasos, etc. També tenen una distribució per pantalles de leds de efectes per concerts i espectacles.
L'inconvenient mes gran per aquesta solució és l'elevat preu dels seus mòduls, El preu del metre quadrat programable ronda els 5.000€ mentre que la solució que es proposa en aquest projecte es possible arribar al metre quadrat de mòduls de led per no més de 600€.

Per lo tant , en aquest projecte s'intenta arribar a un punt mig entre l'electrònica casual de estar per casa i els mòduls programats professionals de molt preu.

S'ha de dir que la velocitat de refresc, ni la densitat de píxel és la mateixa de una empresa professional, però es poden aconseguir moltes coses interesants amb aquesta estructura.

3. DESCRIPCIÓ DE TECNOLOGIES

En aquest apartat del projecte ens disposarem a definir i estudiar les tecnologies que farem servir per la realització del projecte. Primerament descriurem el hardware que utilitzarem i després estudiarem els llenguatges de programació que ens ajudaran a dur a terme el projecte.

3.1 Hardware

Tindrem tres mòduls de hardware per estudiar: La placa Raspberry, el microcontrolador muntat d'Arduino i el bus de comunicació I2C.

3.1.1 Raspberry

La Raspberry pi model B+ és una computadora de placa simple, de molt baix cost produïda al Regne Unit i amb l'objectiu d'estimular l'aprenentatge de ciències de la computació a les escoles i promoure petits projectes casolans.

En essència la Raspberry és un ordinador de baix cost integrat en una placa de la mida d'una targeta de crèdit. És també molt atractiva per ser una plataforma oberta que permet diferents tipus de sistemes operatius (SO) sobre ella. Les distribucions més comunes són les de Linux i les més utilitzades són Raspbian.

A l'actualitat existeixen cinc models de Raspberry: RPi 1 model B+, RPi 2 model B+, RPi 3 model B+, RPi zero i RPi 1 model A+. En aquest projecte optarem pel model RPi 3 model B+.

Raspberry Pi 3 B+: Aquest model (figura) ha estat el més venut i utilitzat i té les següents característiques:

- 542MB de memòria RAM
- Port Ethernet 100Mb
- GPIO amb 40 pins (l'esquema de pins es manté igual que els 26 el model B antic)
- 4 ports USB (el model anterior només tenia 2)
- Full HDMI port
- Combined 3.5mm audio jack and composite video
- Camera interface (CSI)
- VideoCore IV 3D graphics core
- Micro SD (abans era SD) o més baix consum que el seu predecessor

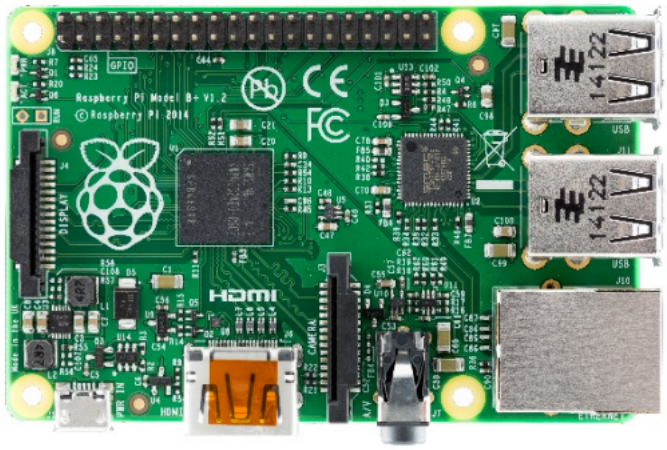


Figura 3

3.1.1.1 SoC (ARM vs X86)

El processador a l'interior de les RPi és un chip multimèdia *Broadcom BCM2835 system-on-chip*(SoC). Això vol dir que la major part dels components del sistema, incloses la CPU y la GPU junts amb l'àudio i el hardware de comunicacions es troben integrats dins d'aquell component únic i ocult, ubicat just sota del processador de memòria RAM de 512MB al centre de la placa.

No és només el disseny tan particular del SoC el que el fa al processador BCM2835 diferent al d'un processador d'un PC o Portàtil. El que també el fa especial és que utilitza una arquitectura de conjunt d'instruccions diferents, coneguda com ARMV. Aquest punt és molt important en el meu projecte ja que las llibreries de Processing d'àudio no funcionen muntades sobre aquesta arquitectura. Això, va fer que des d'un bon principi pensés en treballar des d'un PC pel processament en temps real de l'àudio.

3.1.1.2 CPU

La CPU conte un ARM1176JZFS, amb una unitat de processament amb coma flotant, que funciona 700Mhz i és capaç de suportar un *overclock* de fins a 1Ghz en mode "TURBO", que és una velocitat de processament prou alta per la mida i cost del producte que s'encarrega que el SoC proporcioni més rendiment sense reduir el temps de vida de la placa i sense perdre la garantia de funcionament. La CPU està basada actualment en la versió 6.0 de l'arquitectura ARM, la qual no és suportada per una gran quantitat de distribucions Linux, incloent la més utilitzada Ubuntu. Per Això haurem d' instal·lar Raspbian però ho explicarem més endavant.

3.1.1.3 GPU

La GPU utilitzada és una *Dual Core VideoCore IV Multimèdia Co-Processor*. És capaç de moure continguts amb una gran qualitat visual com *BlueRay*, fent servir

la codificació H264 de fins 40MBits/s. Disposa d'un nucli conegut com 3D amb suport per les llibreries tan conegudes com *OpenGL ES2.0* i *OpenVG*. És capaç de descodificar fins a 1080p a 30fps.

3.1.1.4 RAM

La memòria RAM és de 512 MB del tipus SDRAM (en el seu model B), en un únic mòdul, el qual funciona a una velocitat de 400Mhz en el seu mode normal i pot arribar a una velocitat de 600Mhz en la seva versió mode "TURBO"

3.1.1.5 Emmagatzematge

La RPi no conte un disc dur tradicional muntat sobre la placa base, per corregir això disposa d'un lector/ranura per memòries en forma de targeta SD, un sistema de memòria en estat sòlid. L'arrencada del sistema es realitzarà des de la pròpia targeta SD. Aquest fet provoca que al tenir que contenir tot el sistema operatiu, és necessari que la targeta de memòria sigui de almenys 2GB de capacitat per guardar tots els arxius requerits. Tot i així és recomanable que la targeta tingui més capacitat per albergar programes dins.

Estan disponibles Targetes SD amb el sistema operatiu pre-carregat a la tenda oficial de la Raspberry Pi. Però en el nostre cas obtindrem una targeta SD buida i l'instal·larem (*flashejar*) el S.O. per arrancar el sistema

3.1.1.6 Sortides de Vídeo

Per la sortida de vídeo, la RPi posseeix un connector HDMI (rev 1.3 i 1.4) i una interficie DSI per panells LCD.

La millor qualitat d'imatge es pot obtenir fent servir el connector HDMI (*High Definition Multimedia Interface*). Aquest port proporciona una connexió digital

d'alta velocitat per mostrar imatges de píxels perfectes tant en monitors de PC com en una televisió d'alta definició. Al utilitzar aquest port HDMI, la Raspberry pot desplegar imatges a la resolució de 1920x1080 Full HD. A aquesta resolució, el detall sobre les pantalles és significativament de molt alta qualitat.

L'altre sortida de vídeo que te la RPi és la coneguda com "Display Serial Interface"(DSI), que es fa sevir als monitors de pantalla plana com els de las tauletes interactives o els coneguts com *smartphones*.

3.1.1.7 Bus USB

El model B+ posseeix 4 ports USB 2.0 (via hub USB integrat).Aquest bus es pot fer servir com a serial port connectant un dispositiu extern com l'Arduino.

3.1.1.8 Targeta de Xarxa

Tot i que quan el projecte estigui finalitzat i el prototip estigui tancat no es requerirà connexió a internet, ens serà molt útil per configurar-la i treballar amb ella.

Tenim a la nostre disposició un connector de tipus RJ-45 connectat a un circuit integrat lan9512 –jzx de SMSC que ens proporciona connectivitat de 10 a 100 Mbps.

Els models mes actuals d'aquets dispositius, en particular la RPi 3 B+, contenen un dispositiu de controlador *wifi* del tipus 802.11n *Wireless* LAN.

3.1.1.9 Energia i alimentació

La placa no conté un boto d'apagat o d'encès, amb la qual cosa, l'energia li arriba mitjançant un connector del tipus microUSB estàndard de 5V. El consum de la placa es de 700mA(3,5W).

Molts carregadors dissenyats per *smartphones* funcionaran amb la Raspberry Pi, però no tots ja que alguns nomes subministren fins a 500mA i com ja he dit abans la Raspberry necessita 700mA.

3.1.2 Arduino

És un producte distribuït per una empresa dels Estats Units que consta d'una part hardware i una altre de software. La part hardware està composta per circuits impresos i un micro controlador i el software està lligat a un entorn de desenvolupament (IDE).

La placa consisteix bàsicament en un microcontrolador del tipus *Atmel AVR* i ports d'entrada i sortida, tan analògics com digitals que serveixen per connectar-se amb perifèrics o plaques d'expansió que amplien el rang de projectes que es poden realitzar. Així mateix, també posseeix un port de connexió USB des del que es pot alimentar la placa i establir la connexió amb l'IDE de Arduino i bolcar els programes.

Per altre banda, el software consisteix en un entorn de desenvolupament basat en Processing i un llenguatge de programació basat en *Wiring*, així com el carregador de posada en marxa (*bootloader*) que s'executa en la placa. El microcontrolador de la placa es programa mitjançant un computador, fent servir una comunicació serial mitjançant un convertidor de nivells RS-232 a TTL serial.

Les característiques compartides per la gran majoria de plaques de programació Arduino són:

- El microprocessador ATmega328

- 32 kbytes de memòria Flash
- 1 kbyte de memòria RAM
- 16 MHz
- 13 pins per a entrades/sortides digitals (programables)
- 5 pins per a entrades analògiques
- 6 pins per a entrades analògiques (sortides PWM)
- Completament autònom: Una vegada programat no necessita estar connectat al PC
- Microcontrolador ATmega328
- Voltatge d'operació 5V
- Voltatge d'entrada (recomanat) 7-12 V
- Voltatge d'entrada (límit) 6-20 V
- Digital I/O Pins 14 (amb 6 sortides PWM)
- Entrades analògiques Pins 6
- DC corrent I/O Pin 40 mA
- DC corrent 3.3V Pin 50 mA
- Memòria Flash 32 KB (2 KB per al bootloader)
- SRAM 1 KB
- EEPROM 512 byte
- Velocitat de rellotge 16 MHz

Com he descrit abans, Arduino, no només són components elèctrics, ni una placa de circuits, sinó que a més d'això, també és una plataforma que combina això amb un llenguatge de programació propi que serveix per controlar els diferents sensors i ports que es troben dins de la placa . Això ho podem realitzar per mitja d'instruccions i paràmetres que nosaltres establirem al connectar la placa al usb del nostre PC.

Aquest llenguatge que opera dins del nostre Arduino es diu *Wirirng*, i està basat en la plataforma de Processing i més a baix nivell pel famós llenguatge de programació C/C++ que és l'elegit sempre per ensenyar programació a alumnes de nivell superior en estudis de robòtica o ciències de la computació, gràcies a que és molt fàcil d'aprendre i dona suport a qualsevol necessitat de computació.

Del llenguatge C neixen molts altres que són molt utilitzats en l'àmbit de la Enginyeria i el desenvolupament, com són el *C#, Java, BASIC, Php,Python, Javascript, Perl* entre molts d'altres. Per tant, Arduino és capaç de suportar varis llenguatges de programació d'alt nivell derivats de C, fent d'això una avantatge pels enginyers que treballen en dos o més entorns de desenvolupament de programació.

Per poder treballar des del nivell de programació del processador, és obligatori descarregar-se el software que inclogui las llibreries necessàries per poder utilitzar el llenguatge de manera completa. Una altre avantatge és que aquest software es pot descarregar des del lloc web oficial de l'empresa, ja que opera sota llicència lliure i està disponible a tots els públics. La seva versió mes recent per tots els sistemes operatius es la versió Arduino 1.0.3

En el cas del nostre projecte escollirem la placa Arduino UNO entre totes les demés per la seva comoditat i pell seu preu. També cal dir que és la més comú de totes.

3.1.2.1 Alimentació i potencia(USB)

Cada placa Arduino necessita una forma d'estar alimentada elèctricament per poder fer funcionar el programa que s'ha bolcat des de l'ordinador. Aquesta alimentació pot ser de dues maneres. La primera manera de donar-li energia a la placa és mitjançant un connector del tipus jack mini que contingui 5V a traves d'un transformador.

L'altre forma i també la més comú, és alimentar-la mitjançant un cable USB. Aquesta és la més normal ja que la gravació dels algoritmes que tindrà que processar el microcontrolador dins el nostre programa s'envien mitjançant aquest cable. Com és molt comú equivocar-se i tenir que torna a bolcar el programa, s'utilitza molt més aquest sistema.

És important anar amb compte si s'alimenta mitjançant un mini jack, perquè es subministren més de 10 volts i es pot trencar la placa Arduino. La tensió recomanada és entre 6 i 12 volts.

3.1.2.2 Pins(5V, 3.3V, GND, Analògic, Digital, PWN, AREF)

Els pins de la placa Arduino UNO és per on es connecten els cables d'un circuit o d'una plataforma externa que es comunicarà amb el microcontrolador. L'Arduino UNO té varis tipus diferents d'entrades i sortides que s'utilitzen per coses diverses.

***GND*:** Es l'abreviatura de "terra" en Angles. Hi han diferents pins amb la funció GND en l'Arduino, qualsevol pot ser utilitzat per connectar a terra el circuit implementat i així tenir una referència única pels voltatges de tots els circuits.

5V i 3.3V: Aquets pins no són programables, únicament s'encarreguen de subministrar 5V o 3.3 volts de corrent en el punt que es desitgi.

***Analògic*:** Aquest tipus d'entrades i sortides són del tipus analògiques. Aquests pins poden llegir el senyal d'un sensor analògic (com un sensor de temperatura) entre un rang de 0 a 5V i convertir-lo en un senyal digital que es pugui llegir. Són més cars que els digitals i menys abundants.

Digital: Són els pins digitals els que llegeixen un senyal digital en un rang de 0 a 255. Aquests pins es poden utilitzar tan com entrada digital, és a

dir per exemple si es polsa un botó i de sortida, com per exemple encendre un led i controlar la seva brillantor.

***PWM*:** Aquest tipus de pin són normalment del tipus digital I/O però també es poden configurar amb una característica que es diu modulació per freqüència (*PWM*, per las seves sigles en Anglès). Un cop establert aquest estat, aquests pins tenen les característiques d'un pin Analògic.

***AREF*:** Suports de referència analògica. La majoria de les vegades es pot deixar aquest pin sol. A vegades s'utilitza per establir una tensió de referència externa (entre 0 i 5 Volts) como a límit superior pels pins d'entrada analògica.

3.1.2.3 Botó de reinici

Polsant aquests botó es connectarà temporalment el pin de reset a terra i això farà que es reinici qualsevol codi que s'hagi carregat en l'Arduino. Això és molt útil per si el script no es repeteix però es vol provar varies vegades.

3.1.2.4 Indicador LED d'alimentació

Aquest led s'ha d'encendre cada vegada que es connecti la placa Arduino a una presa de corrent. Si aquesta llum no s'encén, hi ha una gran probabilitat que aquesta placa estigui malmesa.

3.1.2.5 Leds RX i TX

TX és l'abreviatura de transmissió i RX és la de recepció. En aquest cas hi ha dos llocs en l'Arduino UNO on apareixen TX i RX, una vegada pels pins digitals 0 i 1 i una altre vegada junt amb els indicadors LED de TX i RX. Aquets leds ens donen indicació visual sempre i quant el nostre Arduino estigui rebent o enviant dades, com per exemple quan estem carregant un programa a la placa.

3.1.2.6 El microcontrolador

El micro el podem identificar molt fàcilment dins de la placa de circuit, és la barra negre amb totes les potes de metall, aquesta és l'aparença del circuit integrat(IC per les seves sigles en anglès). Aquest és pot interpretar com el cervell de la nostre placa.

La principal IC en els arduino és lleugerament diferent depenent del tipus de placa, però generalment és de la línia ATmega de l'empresa ATMEL. Per l'arduino UNO en concret el microcontrolador és el ATmega328 AVR 8-bit . Aquest està basat en els tipus RISC, combinat amb una memòria flash que proporciona la capacitat de llegir mentre escriu. Conte també 1KB de memòria EEPROM, 2KB de SRAM, 23 línies d'Entrada / sortida de propòsit general i 32 registres de procés general.

El dispositiu opera entre 1.8 i 5 volts de tensió. Mitjançant l'execució de poderoses instruccions en un únic cicle de rellotge, el dispositiu arriba a una resposta de 1 MIPS, balancejant el consum d'energia i velocitat de procés.

3.1.2.7 Regulador de voltatge

Això no és una cosa que literalment es pugui interactuar en Arduino, però és potencialment important saber que existeix i que està muntat en un Arduino i per que el podem utilitzar. El regulador de voltatge fa el que exactament diu que fa, controla la quantitat de tensió que li arriba a la placa Arduino. Es pot pensar que és una mena de vigilant, pararà una tensió addicional que podria fer malbé el circuit. Per suposat, té els seus límits, com hem dit abans si es connecta l'Arduino a una tensió superior a 20 volts l'arduino es pot trencar.

L’esquema global dels Pins, entrades, etc queda com la figura 4:

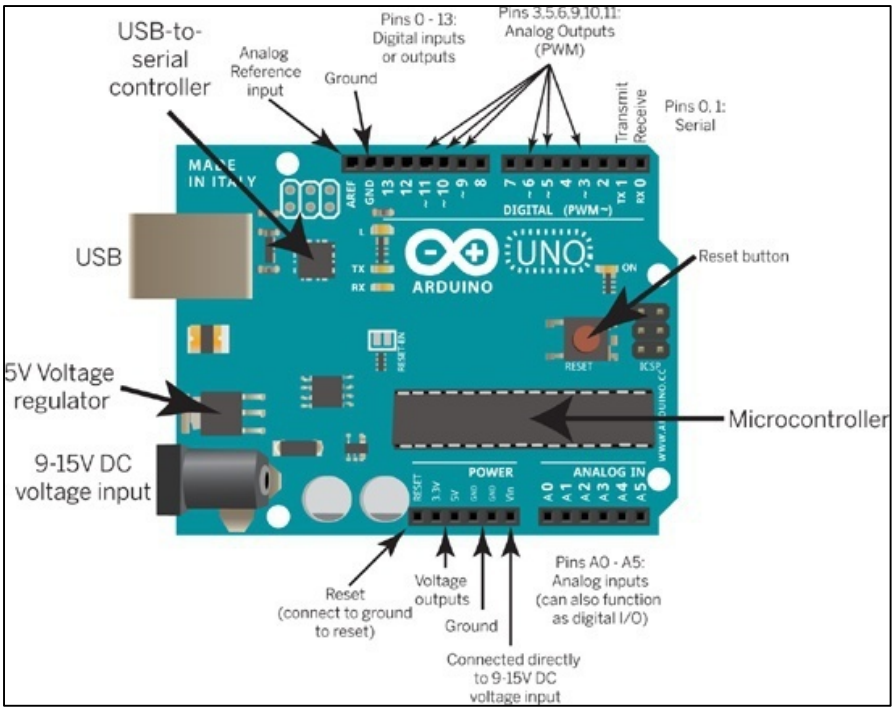


Figura 3. Esquema components Arduino

3.1.3 Bus de comunicació I2C

Aquest bus es defineix per l’abreviatura de Inter-IC, que és un tipus de bus dissenyat per Philips Semiconductors a principis dels anys 80, que s’utilitza per connectar circuits integrats (ICs). El I2C és un bus amb múltiples “mestres”, el que significa que es poden connectar varis chips al mateix bus i que tots ells poden actuar com a mestre, nomes iniciant la transferència de dades. Aquest bus s’utilitza en molts dispositius.

El bus I2C, és un estàndard que facilita la comunicació entre microcontroladors, memòries i altres dispositius amb un cert nivell “d’intel·ligència”. Només

requereix dos lineals de senyal i una referència a massa pels dispositius. Permet un intercanvi d'informació entre molts dispositius a una velocitat bastant gran, uns 100kbits per segon, encara que arriba a punts en els que el rellotge pot tenir un pic de 3.4 Mhz.

La metodologia de comunicació I2C de dades del bus és en sèrie y síncrona. Un dels senyals del bus marca el tempo (polsos de rellotge) i l'altre s'utilitza per intercanviar dades.

Avantatges:

- Requereix pocs cables i connexions
- Disposa de mecanisme per verificar la connexió
- Es eficient d'executar

Inconvenients:

- La seva velocitat és mitjana
- No es full-dúplex
- No hi ha verificació que el contingut del missatge sigui correct

3.1.3.1 Descripció de les senyals

SCL: (*System Clock)* és la línia dels polsos de rellotge que sincronitzen el sistema.
SDA: (*System Data*) és la línia per la que es mouen les dades entre els dispositius.
***GND*:** (*Ground*) Comú de interconnexió entre tots els dispositius enganxats al bus.

Les línies SDA i SDL són del tipus drenatge obert, és a dir, un estat similar al de col·lector obert, però associades a un transistor d'efecte de CAMP(FET). S'hauran de polaritzar en estat alt (connectat a l'alimentació mitjançant resistors del tipus "*pull-up*") el que defineix una estructura de bus que permet connectar en paral·lel múltiples entrades i sortides, com a la figura 5:

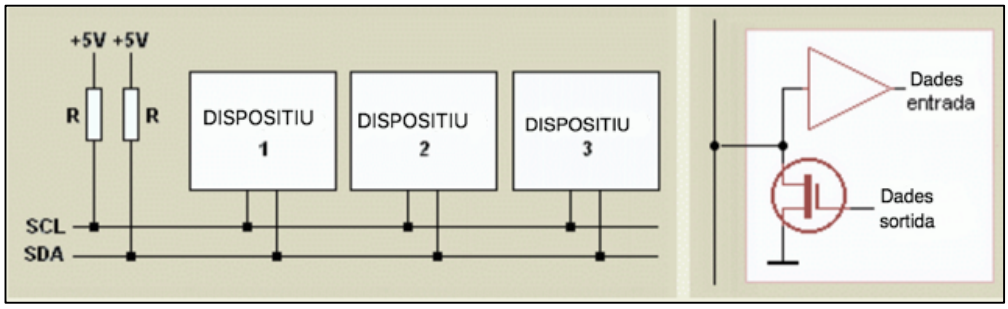


Figura 5. Conjunt de dispositius connectats al bus

El diagrama és suficientment auto-explicatiu. Las dues línies del bus estan en un nivell lògic alt quan estan inactives. En principi, el número de dispositius que es poden connectar al bus no te límits, però s'ha d'observar que la capacitat màxima sumada amb tots els dispositius no superi els 400pF. El valor dels resistors de polarització no és molt estricte i pot anar de 1K8 (1800 ohms) a 47K (47000 ohms). Un valor menor de resistència incrementa el consum dels circuits integrats però disminueix la sensibilitat al soroll i millora el temps dels flancs de pujada i baixada.

3.1.3.2 Protocol de comunicació del bus I2C

Si hi han diversos dispositius connectats sobre el bus, és lògic que per establir una comunicació a traves d'ell, es tingui que respectar un protocol. Diguem , en primer lloc, el més important: existeixen dispositius **mestres** i dispositius **esclaus.** Només els dispositius mestres poden iniciar la comunicació.

La condició inicial, el bus està lliure, és quan ambdós senyals estan en estat lògic alt. En aquest estat qualsevol dispositiu mestre pot ocupar l'espai i establir la condició de ***start*.** Aquesta condició comença quan un dispositiu mestre posa en estat baix la línia de dades(SDA) , però deixant en alt la línia de rellotge (figura 6).

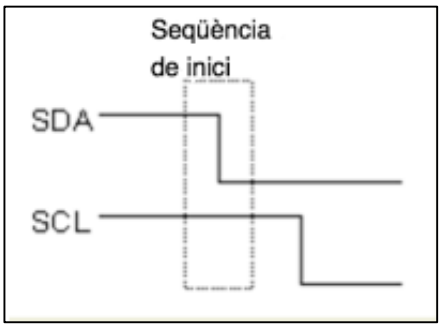


Figura 6. Inici de la comunicació

El primer byte que es transmet després de la condició d'inici conte 7 bits que composen la direcció del dispositiu que es desitja seleccionar per enviar informació, i un altre bit que correspon a l'operació que es vol realitzar, lectura o escriptura.

Si el dispositiu el qual la direcció correspon a la que indicant els 7 primers bits (A0-A6) està present en el bus, aquest contesta amb un bit en nivell baix, que anirà en la posició següent immediatament després del bit 8 que ha enviat el mestre. Aquest bit es reconeix com ACK (figura 7), i indica al dispositiu mestre que l'esclau l'ha reconegut i està en condicions de comunicar-se.

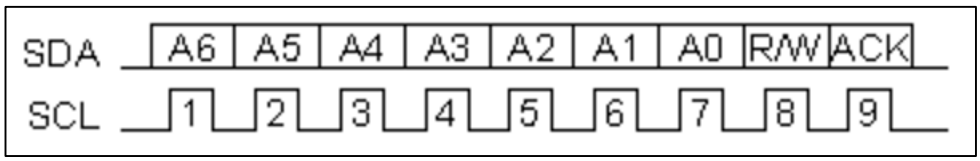


Figura 7. Esquema dels primers 9 bits.

Si el bit de lectura/escriptura (R/W) va ser posat en aquesta comunicació a nivell baix, el dispositiu mestre envia dades a l'esclau. Això és manté contínuament

mentre rebi senyals de reconeixements i el contacte conclou quan s'hagin transmès totes les dades.

En el cas contrari, quan el bit de lectura/escriptura està a nivell lògic alt (lectura), el dispositiu mestre genera polsos rellotge per que el dispositiu esclau pugui enviar dades. Després de cada byte rebut, el dispositiu mestre

El dispositiu mestre en aquest cas deixa el bus lliure, generen una condició de **stop** (figura 8).

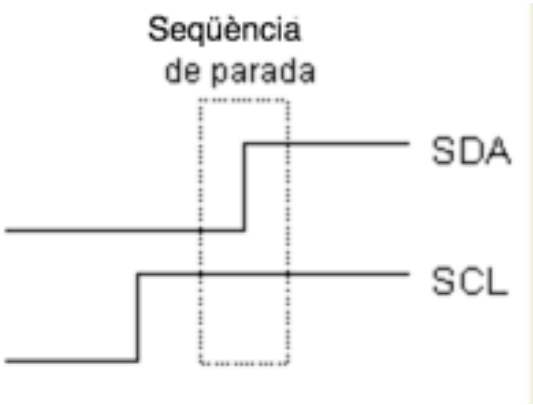


Figura 8. Parada de la comunicació

Si es desitja seguir transmitent, el dispositiu mestre pot generar una altre condició de *start* en comptes d'una de parada. Aquesta nova condició es denomina inici reiterat.

3.1.3.3 Definició de termes empleats en el bus I2C

- ***Mestre*** (*Master*): Dispositiu que determina els temps i la direcció del tràfic del bus. Es l'únic que aplica els polsos de rellotge en la línia SCL. Quan es connecten varis dispositius mestres a un mateix bus, la configuració obtinguda es denomina "multi-mestre".
- ***Esclau*** (*Slave*): Tot dispositiu connectat al bus que no té la capacitat de generar polsos de rellotge. Els dispositius esclaus reben els senyals i el rellotge generat pel mestre.

- ***Bus lliure*** (*Bus Free*): Estat en el que ambdues línies (SDA i SCL) estan inactives, presenten un estat lògic alt. És l'únic moment en que un dispositiu mestre pot començar a fer servir el bus.
- ***Inici*** (*start*): Es produeix quant un dispositiu mestre ocupa el bus, generant la condició. La línia de dades (SDA) pren un estat baix mentre que la línia de rellotge roman en estat alt.
- ***Parada*** (*Stop*): Un dispositiu mestre és el que pot generar aquesta condició, deixant lliure el bus. La línia de dades i rellotge prenen un estat lògic alt.
- ***Dades valides*** (*Valid Data*): Situació present quan una dada que estigui present a la línia SDA és estable al temps que la línia de SCL es manté a nivell alt.
- ***Format de les dades*** (*Data Format*): La transmissió d'una dada a través d'aquest bus consisteix en 8 bits(1 byte). A cada byte transmès al bus, el segueixen un novè pols de rellotge durant el qual el dispositiu receptor del byte ha de generar un altre pols de reconeixement.
- ***Reconeixement*** (*Acknowledge)*: El pols de reconeixement, conegut com ACK, s'aconsegueix col·locant la línia de dades a un nivell lògic baix durant el transcurs del novè pols de rellotge.
- ***Adreça*** (*Adress*): Tot dispositiu dissenyat per funcionar en aquest bus posseeix la seva pròpia i única adreça d'accés, preestablerta pel fabricant. Hi ha dispositius dissenyats per establir externament part de l'adreça d'accés, el que fa que es puguin connectar en un mateix bus dispositius del mateix tipus, sense problemes d'identificació. L'adreça 00 és la denominada d'accés general, és a la que responen tots els dispositius connectats al bus.
- ***Lectura/Escritura*** *(R/W*): Cada dispositiu te una adreça de 7 bits, el vuité bit, el de menys pes, que s'envia a través del mestre durant l'operació d'adreçament indica quina operació s'ha de realitzar. Si aquest bit té un valor alt, significa que l'esclau envia cap al mestre. Si aquest bit és baix, el dispositiu mestre envia a l'esclau.

3.1.3.4 Procés de comunicació

Quan el dispositiu mestre es vol comunicar inicia la seqüencia de *start*. Aquesta seqüencia és una de les dues trames especials del sistema, i són especials perquè és un dels dos únics moments on es permeten canvis a la línia SDA quan la seqüencia de rellotge està alta.

Les dades és transfereixen en seqüencies de 8 bits, un byte. Els bits es col·loquen en la línia de SDA començant per el bit de major pes. Una vegada posat un bit en la línia de dades, es porta la línia de SCL en alt. Cada cop que el dispositiu de recepció rep 8 bits, aquest envia un bit de reconeixement, pel que en realitat cada byte que es transmet es produeixen 9 polsos sobre la línia SCL. Quan aquest bit és un nivell baix, indica que ha rebut les dades correctament i està llest per rebre'n mes. Si pel contrari retorna un bit en alt indica que no es poden rebre més dades i llavors el dispositiu mestre està obligat a enviar una seqüencia de stop.

3.1.3.5 Adreces en el bus I2C

Per establir una adreça en el bus I2C, el més comú és que utilitzin adreces de 7 bits, però també n'hi han amb 10 bits.

Una adreça de mida de 7 bits implica que es poden posar fins a 128 dispositius sobre aquest bus, ja que les adreces poden ser de 0 a 127. Quant s'envien les adreces, la transmissió segueix sent de 8 bits. El bit extra s'utilitza per informar al dispositiu esclau que el dispositiu mestre escriurà o llegirà.

Si es desitja escriure l'adreça 21(HEX), en realitat es deu enviar un 42, que es un 21 desplaçat un bit cap amunt.

3.1.3.6 Programació del bus I2C i seqüencia d'escriptura

Quan el primer dispositiu es posa en seqüencia d'inici els esclaus es posen en alerta a l'espera de transacció. Aquets es queden esperant a veure si la sol·licitud va cap a ells.

Un cop l'adreçament ja s'hagi establert, el que fa el mestre ara és enviar la ubicació interna o número de registres des del que es desitja escriure o llegir. La quantitat depèn òbviament de les característiques tècniques de cada dispositiu.

Un cop el mestre envia l'adreça i el registre intern del dispositiu, pot començar la comunicació de bytes que es vulgui. Es pot anar seguint enviant dades contínuament ja que el dispositiu esclau va incrementant l'adreça de registre intern a mesura que rep dades.

La seqüencia d'escriptura del mestra al esclau queda definida per els següents punts:

1. Enviar una seqüencia de *start*
2. Enviar l'adreça del dispositiu amb el bit de escriptura/lectura
3. Enviar el número de registre intern en el que es desitja escriure
4. Enviar el o els bytes de dades
5. Enviar la seqüencia de stop

3.1.3.7 Ressenya

Els inconvenients pel meu projecte es poden solucionar. El primer perquè el procés de l'àudio es fa en un altre lloc, que és l'únic que podria generar latència. El segon, ja que només haurem d'enviar en una direcció sempre. Y per últim el missatge que s'envia està prèviament processat i analitzat perquè sigui el correcte.

3.2 Software

En aquest apartat estudiarem els llenguatges els quals gracies a ells podrem programar el projecte sencer en el diferents mòduls de hardware. Utilitzarem tres llenguatges diferents, primerament Python i després Processing i ArdunioIDE que són molt similars.

3.2.1 PYTHON

Python és un llenguatge de programació interpretat, el qual té una filosofia que posa especial èmfasi en una sintaxi que afavoreixi un codi llegible i interpretable per a tothom.

Aquest és un llenguatge que es considera poderós per les múltiples aplicacions i plataformes que el desenvolupen i fàcil d'aprendre. Compta amb una estructura de dades eficient i d'alt nivell i un enfocament simple però efectiu a la programació orientada als objectes. La seva sintaxi elegant i senzilla, junt amb la seva naturalesa interpretada fan d'aquest llenguatge, un ideal per desenvolupar moltes aplicacions fàcils i ràpides sobre la majoria de plataformes.

L'intèrpret de Python i la extensa biblioteca estàndard estan a la lliure disposició de tothom en forma binaria i també el codi font per a les principals plataformes des del lloc web oficial. Això indica el caràcter de codi lliure que té aquest projecte. El mateix lloc web també conté informació sobre distribucions, enllaços a mòduls lliures de Python, programes, eines ,etc.

3.2.1.1 Història

Guido Van Rossum va idear el llenguatge Python a finals dels anys 80 i va començar a implementar-lo el desembre de 1989. El febrer de 1991 va publicar la primera versió pública, la versió 0.9.0. La versió 1.0 va haver d'esperar a modificacions de la primera i va sortir el gener de 1994. Molt més tard van arribar les actualitzacions del llenguatge, la versió 2.0 i versió 3.0 es van publicar el 2000 i 2008 respectivament.

El desenvolupament de Python el du a terme un col·lectiu de programadors que treballen sota el paraigües de la "Python Software Foundation", però supervisats pel seu creador. Les versions de Python s'identifiquen per tres números X, Y i Z, els quals:

- X correspon a les grans versions de python(1, 2 o 3), incompatibles entre sí.
- Y correspon a versions importants en les que s'introdueixen novetats en el llenguatge però mantenint la compatibilitat.
- Z correspon a versions menors que es publiquen durant el període de manteniment, en les que només es corregeixen errors i problemes de seguretat.

3.2.1.2 Elements del Llenguatge

Python va ser dissenyat per ser llegit amb facilitat. Una de les seves principals característiques és l'ús de paraules on altres llenguatges utilitzaries símbols. Per exemple per declarar una condició d'operador lògic , es declara *not* i no un símbol del tipus !.

El contingut dels blocs de codi també es diferent, està delimitat per espais i per tabulacions, es coneix com el mètode de sagnat, abans de cada línia d'ordres

del bloc. Python es diferencia així dels altres llenguatges de programació que mantenen la costum de C de declarar els blocs mitjançant un conjunt de caràcters, normalment claudàtors.

3.2.1.3 Variables

Les variables es defineixen de forma dinàmica, el que significa que no es tenen que especificar quin és el tipus abans de fer-la servir, i pot prendre diferents valors en un altre moment, incloent a un tipus diferent al que tenia prèviament declarat. Algunes són:

- ***long:*** número enter
- ***float:*** número decimal
- ***complex:*** número complexe
- ***bool:*** boolea

3.2.1.4 Llistes i *tuplas*

Una llista o una *tupla* és un conjunt ordenat d'elements. Per declarar una llista es fan sevir els claudators i per declarar una *tupla* s'utilitzen els parèntesis. En ambdues formes els elements se separen per comes. Els dos sistemes de emmagatzematge d'elements es poden guardar diferents tipus de variables.

Per accedir a un element en concret de la llista o *tupla* s'utilitza un índex enter, començant per el 0. Es poden fer servir índex negatius per accedir als elements a partir del final.

La diferència entre l'una i l'altre és que les llistes poden canviar el seu contingut durant el temps d'execució, mentre que les *tuplas* no.

3.2.1.5 Diccionaris

Un diccionari és una llista on s'emmagatzemen elements i que tenen un identificador que es coneix com clau. Per declarar un diccionari, s'utilitzen els símbols de {}. Els elements van separats per comes, on cada element té l'estructura del tipus "*clau:valor*".

Els diccionaris són mutables, és a dir, es pot canviar el contingut d'un valor en temps d'execució. En canvi, les claus d'un diccionari han de ser immutables. El valor associat a una clau pot ser de qualsevol tipus de dades, fins i tot un altre diccionari.

3.2.1.6 Funcions

Les funcions són petits programes que executen instruccions fora del *main* per retornar una variable o modificar-ne alguna. Les funcions a Pyhton es defineixen amb la paraula clau *def*, seguida del nom de la funció i els paràmetres que entren a la funció.

Una altre forma de escriure una funció, però molt poc utilitzada, és amb la paraula clau lambda. El valor retornat per la funció serà donat amb la instrucció *return*.

3.2.1.7 Classes

És un objecte en qual definim els seus atributs, funcions i mètodes. És un dels principals mètodes utilitzats a la programació. Les classes a Python es defineixen amb la paraula clau *class*, seguida del nom de la classe i si ve d'una altre classe, el nom d'aquesta.

En Python 2.x és recomanable que una classe hereti les característiques de "*object*". En les classes, un mètode equival a una funció i un atribut és el mateix

que una variable. Els atributs que es desitgin que siguin accessibles des de fora de la classe es tenen que declara fent servir el "*self*", davant del nom.

3.2.1.8 Condicionals

Una sentència condicional, s'executa el seu bloc de codi intern només si es compleix certa condició declarada al principi del bloc. Es defineix fent servir la

3.2.2 PROCESSING

3.2.2.1 Descripció

És un llenguatge de programació i entorn de desenvolupament integrat de codi obert basat en Java, de fàcil utilització i que serveix com a mitjà d'ensenyament i producció de projectes multimèdia i interactius de disseny digital.

Va ser creat per Ben Fry i Casey Reas a partir de reflexions en un grup de projectes del MIT Media Lab dirigit per John Maeda. Es distribueix sota la llicencia de GNU GPL.

Breu resum de JAVA

Java és un llenguatge de programació de propòsit general orientat a objectes desenvolupat per Sun Microsystems. També es pot dir que Java és una tecnologia que no només redueix el llenguatge sinó que proveeix una maquina virtual que permet executar el codi compilat per *JAVA*, sigui quina sigui la plataforma que existeix.

El llenguatge s'inspira en altres llenguatges i tenen en comú:

- Sentències comuns de C i C++
- Definicions de interfaces semblants a la de Objective C
- Gestió de emmagatzematge automàtic semblant a Lisp

Basat en el llenguatge C++ però on s'eliminen moltes de les característiques de OOP que s'utilitzen esporàdicament i creen freqüents problemes pels programadors. Aquesta delimitació de causes d'error i problemes de manteniment facilita i redueix el cost del desenvolupament del software.

El llenguatge Java es va crear amb cinc objectius principals:

1. Hauria de fer servir el paradigma de la programació orientada a objectes.
2. Hauria de permetre l'execució d'un mateix programa en múltiples sistemes operatius.
3. Hauria d'incloure per defecte suportar treballar en xarxa.
4. Hauria de dissenyar-se per executar codi en sistemes remots de forma segura.
5. Hauria de ser fàcil d'utilitzar i prendre el millor d'altres llenguatges orientats a objectes, com C++.

Estructura base dels Scripts de Processing:

```
Void setup()
{
     declaracions;
}
void Draw()
{
     declaracions;
}
```

3.2.2.2 Orientació a objectes

La primera característica, orientada a objectes, es refereix a un mètode de programació i al disseny del llenguatge. Encara que hi ha moltes interpretacions per OO, una primera idea es dissenyar el software de forma que els diferents tipus de dades que utilitzin estiguin units a les seves operacions.
Així, les dades i el codi (funcions o mètodes) es combinen en entitats anomenades objectes. Un **OBJECTE** pot veure com un paquet conté el comportament (codi), i l'estat (dades). El principi és separar allò que canvia de les coses que romanen inalterables.

Un altre de les grans fites de la programació orientada als objectes és la creació d'entitats més genèrica que permetin la reutilització del software entre projectes, una de les premisses fonamentals de la enginyeria de Software. Un objecte genèric hauria en teoria, de tenir els mateixos conjunts de comportaments en diferents projectes.

3.2.2.3 Independència de la plataforma

Un altre característica, la independència de la plataforma, significa que els programes escrits en llenguatge Processing poden executar-se a tots els tipus de hardware que tinguin instal·lat un compilador JAVA. Això significa que és capaç d'escriure un programa una vegada i que es pugui executar en qualsevol dispositiu.

Per poder dur a terme això, es compila el codi font escrit en llenguatge Java, per generar un codi conegut com "*bytecode*", específic de Java. Aquesta peça està a mig camí entre el codi font i el codi màquina que entén el dispositiu destí. El *bytecode* és executat per la màquina virtual de Java que interpreta i executa el codi. A més s'accedeixen a llibreries pròpies de cada dispositiu com poden ser els gràfics.

3.2.2.4 Expressions

Les expressions són un conjunt d'elements que són avaluats per retornar un resultat. Els *tokens* són l'element més petit d'un programa que és significatiu i entès pel compilador. En Processing els *tokens* es divideixen en cinc categories que són:

3.2.2.4.1 Identificadors

Són les representacions que es dona als noms que s'assignen a les variables, classes, paquets, mètodes i constants en el codi de Java per que el compilador els identifiqui i el programador els pugui entendre. En Processing com a en Java, els identificadors es poden diferenciar entre majúscules o minúscules.

3.2.2.4.2 Paraules clau

Són els identificadores reservats per Java per complir amb un objectiu específic en el codi i el compilador. S'utilitzen de forma limitada i en casos específics. Tenen unes característiques preestablertes i són les encarregades d'establir les variables. Uns exemples de paraules clau son: *Char*, *class*, byte, *for* etc

3.2.2.4.3 Literals i constants

Els literals són sintaxis per assignar valors a una variable, és a dir, el valor que pot prendre una variable, també és un valor constant que pot ser del tipus numèric. Les constants són variables que tenen un valor fixe i no poden ser modificades en el transcurs de l'execució del codi, aquestes es declaren per mitjà dels modificadors.

3.2.2.4.4 Operadors

Són els elements que ens indican una avaluació que s'aplica a un objecte o dada, sobre un identificador o constant. Un exemple d'operador pot ser la suma, resta o multiplicació.

3.2.2.4.5 Separadors

S'utilitzen per indicar-li al compilador de processing on s'ubiquen els elements del codi, els separadors que admet *JAVA* son: {},:;

3.2.2.5 Operadors

Els operadors són aquells que després de realitzar una operació retornen un resultat, aquets es poden caracteritzar pel número d'operadors, el tipus d'elements en el que s'operen i el resultat que generen.

El número d'operadors pot ser de dos tipus unaris o binaris. Els unaris són aquells que només necessiten d'un element per retornar un valor, mentre que els binaris necessiten de dos o mes elements.

Els operadors són una part principal en les expressions, el tipus i la forma d'ús és fonamental a l'hora de programar, però pel seu ús s'ha de tenir en compte una sèrie de normes.

3.2.2.6 Sentències

Les sentències són una representació d'una seqüència d'accions que es realitzen en processing. La clau fonamental de les sentències és el punt final que indica que ha finalitzat la mateixa, i que es pot continuar amb la següent. L'indicador per canviar de sentencia és el signe de punt i coma (;).

Les sentències es poden classificar per assignació, de bucles, condicionals i de salt. Les sentències es conformen comunament per una instància, i un operador, un exemple és la sentència d'assignació que es conforma per una instància de variable el signe d'assignació i una expressió:

```
int variable = 12+2;
```

- Les **sentències d'assignació** són aquelles a les que s'assigna un valor a una variable o constant.

- Les **sentències condicionals** són les que expressen una condició per definir el flux d'execució del programa, entre elles tenim el if-else y switch.
- Les **sentències de bucles** s'encarreguen de realitzar una acció durant un cert temps donat o fins que es compleixi una condició preestablerta, entre elles tenim el while, do-while.

- Les **sentències de salt** porten el compilador a un punt específic del programa o cap a la següent sentencia d'execució, entre elles tenim break, continua i return.

3.2.2.7 Conversió de tipus

En alguns casos sol ser necessari convertir un tipus de dada a una diferent, això es coneix com la conversió de tipus, modelat, així d'aquesta forma podem realitzar les operacions necessàries sobre el valor que es desitja convertir.

S'ha de tenir en compte el tipus de dada que es va a convertir, ja que si s'ha de convertir una dada que tingui una quantitat de bit inferior a la que estava, aquesta dada tindrà una pèrdua d'informació, per exemple si desitgem convertir un long a un enter (*int*), el compilador eliminarà els primer 32 bits del long per ajustar-lo a l'enter.

3.2.3 Arduino IDE

3.2.3.1 Descripció

Arduino és una plataforma d'electrònica oberta per a la creació de projecte i prototips basada en software i hardware lliure, flexible i fàcil de fer servir. El microcontrolador a la placa Arduino es programa mitjançant el llenguatge de programació Arduino (basat en *Wiring*) i el entorn de desenvolupament Arduino (basat en Processing).

L'entorn de desenvolupament d'Arduino és propi i està basat en Java, gràcies a això és multi plataforma i es pot descarregar molt fàcilment des del lloc web oficial d'Arduino. Les instruccions d'instal·lació són molt senzilles i venen descrites en la mateixa web.

3.2.3.2 Estructura de Programa

L'estructura bàsica del llenguatge de programació d'Arduino és bastant simple i es composa de com a mínim dues parts. Aquestes dues parts necessàries, o funcions, estan delimitades com a blocs amb declaracions, estaments o instruccions.

Estructura base:

```
Void setup()
{
     declaracions;
}
void loop()
{
     declaracions;
}
```

***Setup*()** és la part encarregada de recollir la configuració del nostre programa, i ***loop*()** és la que conté el programa que s'executarà cíclicament (d'aquí el seu nom de *loop-bucle* en anglès). Les dues funcions són imprescindibles perquè el programa treballi correctament.

La funció de configuració deu contenir la declaració de les variables. És la primera funció que executa el programa, i només s'executa una vegada. S'utilitza per configurar o inicialitzar els modes del pin (entrada/sortida), configuració de la comunicació en sèrie i altres.

La funció bucle és la que conté el codi que s'executarà contínuament un cop traslladat a la placa. Aquesta funció és el nucli de tots els programes d'Arduino i la que realitza la major part del treball.

3.2.3.3 Funcions

Una funció és un bloc de codi que té un nom i un conjunt d'instruccions que són executades quan es crida la funció. Són funcions el *setup*() i el *loop*() de les quals ja he parlat. Les funcions d'usuari poden ser escrites per realitzar tasques repetitives i per reduir la mida d'un programa.

Les funcions es declaren associades a un tipus de valor "*type*". Aquest valor serà el que retorna la funció un cop acabi d'avaluar totes les sentències de la mateixa, per exemple 'int' s'utilitzarà quan la funció retorni una dada del tipus numèrica enter. Si la funció no retorna cap valor sinó que modificarà alguna variable existent o similar, llavors es col·locarà davant de la declaració de la funció la paraula 'void', que significa buida.

Després de declarar el tipus de dada que retorna la funció, s'ha d'escriure el nom de la funció i entra parèntesis, s'escriuran si és necessari els paràmetres que

necessita la funció perquè s'executi correctament. L'estructura llavors quedarà així.

```
type nomFunció (paràmetres)
{
     instruccions;
}
```

3.2.3.4 Blocs

Els blocs d'instruccions, ja siguin en funcions principals o funcions d'usuari, es tanquen amb claus {}. Les claus serveixen per definir el principi i el final d'un bloc de sentències. S'utilitzen també per delimitar condicions com són ara el *if, while* etc.

Una clau d'obertura de sentències "{" sempre ha d'anar seguida per una clau de tancament "}", sinó es així el programa donaria errors.

El programa d'Arduino IDE i el seu entorn de programació inclou una eina de gran utilitat per comprovar el total de claus que hi ha. Només s'ha de clicar una clau d'obertura de sentències i immediatament es marca la clau corresponent a tancament d'aquell bloc. Aquesta eina ens ajuda a saber on està delimitat el bloc que interessa.

3.2.3.5 Sentències o instruccions

Aquestes frases són les encarregades de donar les instruccions necessàries que s'executaran en el nostre programa. El punt i coma ";" s'utilitza per separar instruccions en el llenguatge de programació d'Arduino. També s'utilitza per separar els elements en una instrucció del tipus "bucle for".

També es poden posar comentaris a les sentències a moda de que sigui més intel·ligible el codi que creem o per posar alguns apunts puntuals. Aquestes sentències són ignorades pel programa. Comencen per /* i acaben per */, i poden ocupar varies línies.

Degut a que els comentaris són ignorats pel compilador i no ocupen espai a la memòria de placa Arduino, poden ser utilitzats molts cops.

També es pot declarar en una lineal de comentaris. Aquesta, comença quan s'escriu // i acaba a la següent línia de codi. Al igual que els comentaris en bloc , aquestes línies són ignorades pel programa i no ocupen espai a la memòria. Una línia de codi s'utilitza normalment per donar informació de la mateixa línia i el bloc de comentaris és més explicació genèrica de la funció o programa.

4. DESENVOLUPAMENT DEL PROJECTE

En aquest apartat explicaré els passos que he seguit per acoblar el hardware i com he dissenyat el software per implementar el meu projecte. Com a la descripció, aquest procés es dividirà en dos grans blocs; El Hardware i el Software.

4.1 Hardware

En aquest punt descriurem com instal·lar el sistema operatiu en la Raspberry i preparar la per utilitzar la matriu led a traves dels seus Pins GPIO. També descriurem el procés de gestió del bus I2C a través de la pla Arduino.

4.1.1 Raspberry

En aquest apartat descriurem els passos que hem de fer per configurar la placa Raspberry i preparar-la perquè treballi amb el perifèric d'Adafruit.

4.1.1.1 Instal·lació de Raspbian

La instal·lació de Raspbian la faré mitjançant NOOBS, que és l'instal·lador oficial pels sistemes operatius de Raspberry. Existeixen dues opcions d'adquirir NOOBS: Comprant una targeta SD que l'inclogui la pàgina web oficial o instal·lar-ho en una SD que es compri buida.

Jo he triat la segona opció per ser la més còmode i més ràpida de realitzar. Dins de cada una de les opcions apareixen dues maneres d'instal·lar el sistema operatiu: **NOOBS *offline*** i **NOOBS LITE.** La primera opció requereix d'un fitxer de major mida en el moment de la descàrrega, però seran necessàries menys descàrregues en el procés d'instal·lació. Aquets dos fitxers es poden descarregar de la pàgina web oficial de Raspberry tal i com mostra la figura .

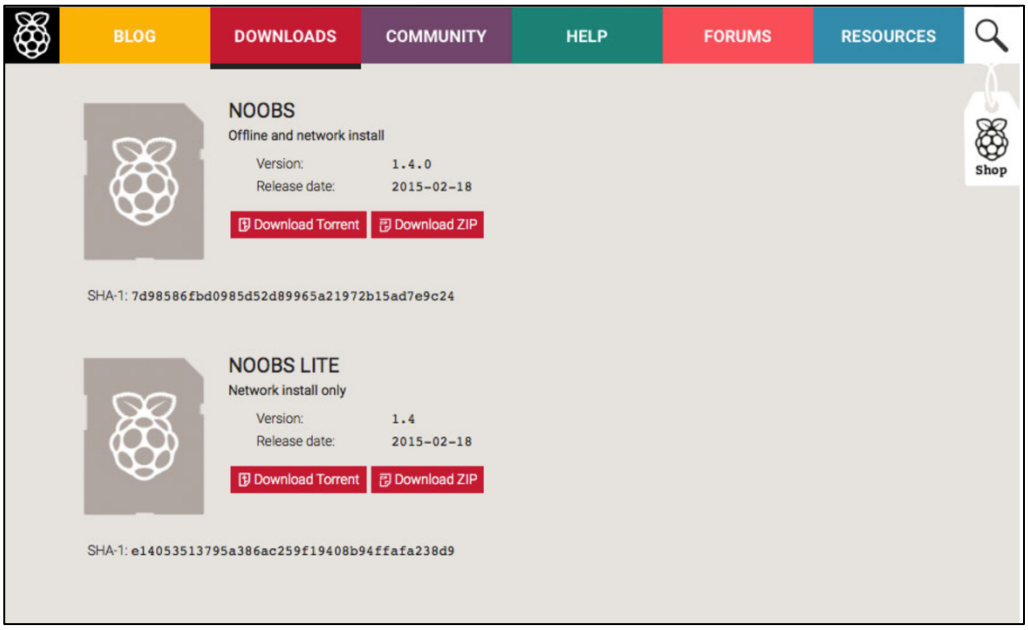


Figura 9. Pagina oficial de Raspberry

Després de descarregar el sistema operatiu s'hauran de seguir el següents passos:

1. Inserir la targeta SD en el ordinador.
2. Formatar la targeta SD per això farem servir les eines del IOS de mac per formatar disc durs.
3. Descomprimir NOOBS(.zip) a la targeta SD formatada.

Una vegada s'ha descomprimit NOOBs a una targeta SD es procedirà a instal·lar el sistema operatiu. Posarem la targeta SD en la Raspberry, seguirem els passos següents d'instal·lació:

1. Connectar el cable HDMI, el ratolí, el teclat i l'alimentació. També serà necessari connectar el cable de xarxa *Ethernet* o configurar la xarxa *Wifi* de casa. En el meu cas utilitzant la Raspberry pi 3 és possible configurar una xarxa *Wifi* particular.

2. Una vegada tot connectat, la targeta arranca per primera vegada i crea les particions necessàries.

3. A continuació, apareix un menú en el qual es mostren diferents sistemes operatius que es poden instal·lar (figura 10). En la part inferior d'aquest menú es pot seleccionar l'idioma.

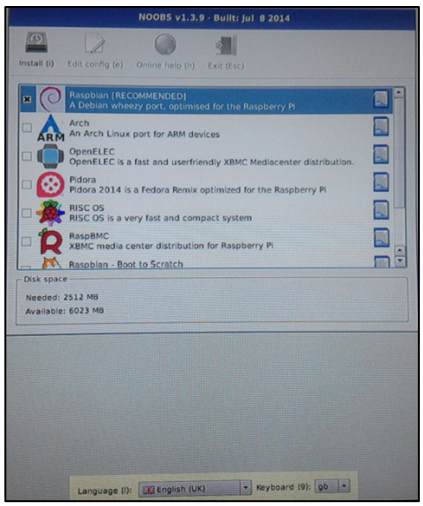


Figura 10. Menú de setup

En la finestra se seleccionen els sistemes operatius que es desitgen instal·lar simultàniament a la targeta SD. A la part inferior es mostra l'espai disponible. Es recomana instal·lar únicament Raspbian per no sobrecarregar la targeta SD.

4. Una vegada confirmat el començament a traves del botó "*install*" apareixerà un missatge per indicar que la targeta SD serà esborrada. Per continuar s'haurà de prémer "*Yes*". Apareixerà una finestra mentre es realitza l'acció.

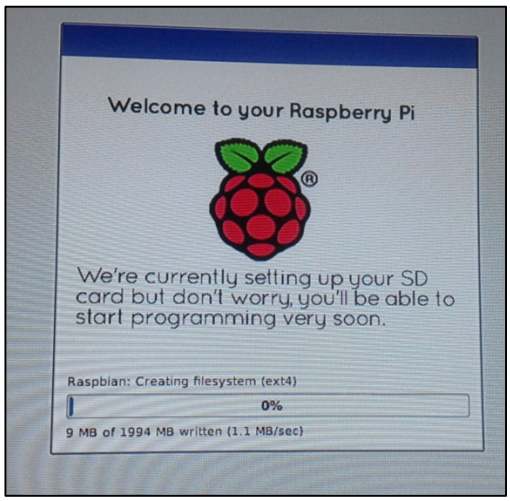


Figura 11. Proces d'instalació del SO

5. Un cop finalitzat aquest procés, es notificarà amb una finestra a la que ens ha d'aparèixer un OK que tindrem que prémer conforme la configuració és correcte. Després es procedirà amb la primera arrencada. L'usuari i contrasenya per defecte son: USUARI:pi CONTRASENYA: raspberry

6. Després de la instal·lació s'obre l'eina raspi-config que permet configurar Raspbian (figura 12). Es pot accedir a aquest menú en qualsevol moment escrivint a la línia de comandes de la terminal : sudo raspi-config.

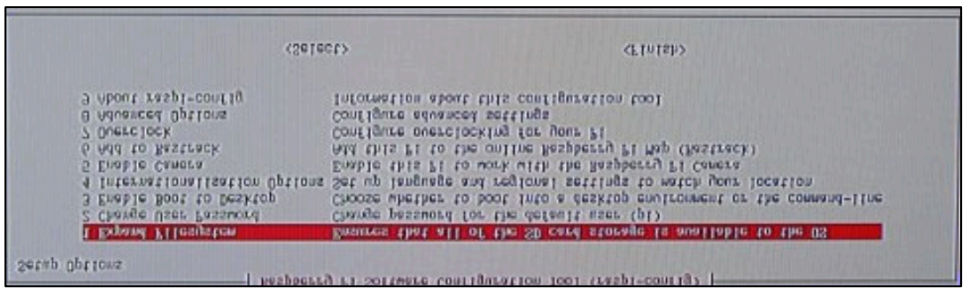


Figura 12. Procés de configuració del SO

6.1 En primera instancia, s'ha de seleccionar la primera opció *Expand Filesystem,* per aprofitar tota la targeta SD i que ocupi tota la seva capacitat. A l'acabar aquest procés, el sistema es reiniciarà.

6.2 A continuació, es recomana canviar la contrasenya mitjançant la segona opció *Change User Password.*

6.3 Si es vol que el sistema arranqui directament a l'escriptori i no mitjançant la línia de comandes (terminal), s'ha d'utilitzar la tercera opció, E*nable Boot To Desktop or Scratch*.

6.4 Amb la quarta opció, *International Options*, és possible configurar la zona horària i el llenguatge.

6.5 Al finalitzar la configuració, s'ha de prémer el boto de "Finish" i el sistema es reiniciarà.

Arribats a aquest punt el sistema operatiu estarà instal·lat.

4.1.1.2 Estudi de pins i punts de soldadura

Per preparar la matriu de la pantalla de leds *Adafruit RGB matrix*, necessitarem primer estudiar el seu comportament amb el GPIOS i després les eines necessàries per soldar la matriu: Estany i un Soldador.

4.1.1.2.1 Estudi de Pins

Pel control de la matriu de leds farem servir els GPIO de la Raspberry. La Raspberry compte amb pins de comunicació externa i referències(GND). Pel control de la pantalla de leds no els farem servir tots, però els que farem servir seran:

- PIN GPIO 5: Aquest pin controla els leds vermells de la meitat superior de la matriu.

- PIN GPIO 13: Aquest pin controla els leds verds de la meitat superior de la matriu.

- PIN GPIO 6: Aquest pin controla els leds blaus de la meitat superior de la matriu.

- PIN GPIO 12: Aquest pin controla els leds vermells de la meitat inferior de la matriu.

- PIN GPIO 16: Aquest pin controla els leds verds de la meitat inferior de la matriu.

- PIN GPIO 23: Aquest pin controla els leds blaus de la meitat inferior de la matriu.

- PIN GPIO 4: Aquest pin controla el nivell de la il·luminació de tots els leds.

- PIN GPIO 17: Aquest pin és el clock que controla la velocitat de bits.

- PIN GPIO 21: Aquest pin controla el bloqueig de dades.

- PIN GPIO 22:Control de multiplexació del led

- PIN GPIO 26: Control de multiplexació del led

- PIN GPIO 27: Control de multiplexació del led

- PIN GPIO 20: Control de multiplexació del led

4.1.1.2.2 Acoblament i soldadura

Un cop estudiat el comportament dels pins i la seva connexió amb la matriu d'Adafruit, es pot continuar amb l'acoblament del hardware.

1. Es comença per connectar la capçalera de pins de 2x20 en una Raspberry Pi, això el mantindrà estable, mentre es soldin els pins.

2. Es començarà soldant el dos punts situats als extrems de la primer lineal. Això farà que la matriu s'equilibri (figura 13).

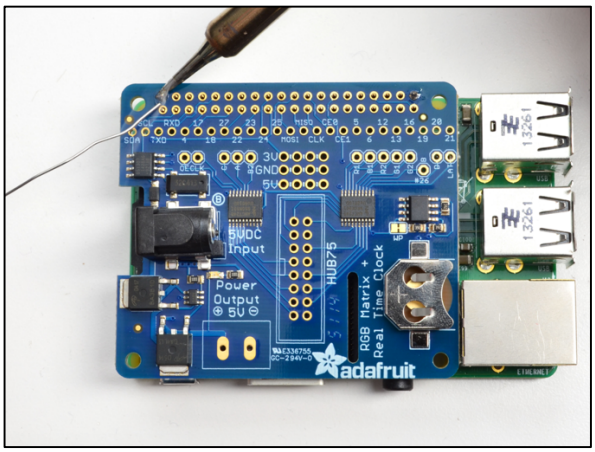


Figura 13. Procés de soldadura dels Pins

3. A partir d'aquestes soldadures es pot soldar la fila sencera i repetir el mateix procediment per la línia inferior.

4. Comprovar que els punts de soldadura són brillants i que no se surten del seu radi. Si es veu alguna malformació es recomana dessoldar tota la matriu i tornar a repetir el procés ja que sinó podria esdevenir en una mala connexió i la matriu led no es connectaria bé.

5. Un cop soldada la connexió amb el GPIOS de la Raspberry, es pot extreure la matriu de la Raspberry i és dona la volta. A les ranures centrals acoblarem el connector BCM de 2x8. Un cop acoblat procedirem a la soldadura igual que el 2x20.

6. Per últim procedirem a la connexió del sistema d'alimentació de la placa matriu. Per això connectarem el pins adients a les últimes dues ranures de la matriu i les soldarem.

4.1.2 Arduino + Bus I2C

La implementació hardware d'aquest apartat és molt senzill per això he integrat les dues tecnologies en un mateix capítol.

Arduino dona suport i disposa de la capacitat de comunicació I2C per hardware vinculats directament a certs pins físics de la placa. També és possible programar el software perquè qualsevol altre grup de pins de la placa actuï com un bus I2C i comunicar-se entre ells, l'inconvenient és que la velocitat de transmissió seria molt més baixa.

Els pins als que està associat el bus I2C varia d'un model d'Arduino a un altre. La següent taula mostra la disposició en els principals models de la marca:

MODEL	SDA	SCK
Uno	A4	A5
Nano	A4	A5
Mini Pro	A4	A5
Mega	20	21

Figura 14. Taula de Pins I2C en diferents Arduinos

En el nostre projecte hem optat per fer servir el model més comú i fàcil de trobar que és l'Arduino UNO (Figura 14). Per tant és tan fàcil com connectar els pins analògics A4 d'una placa amb la A4 de l'altre , després els pins analògics A5 de la placa transmissora amb el A5 de la placa receptora. Per últim connectarem

les dues referències a terra GND de les dues plaques per que tinguin la mateixa referència a l'hora de generar els polsos de transmissió.

La connexió final quedaria de la següent figura:

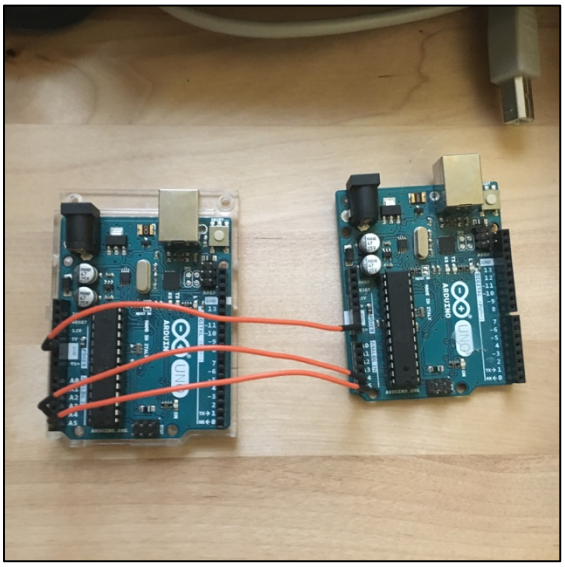


Figura 15. Connexió bus I2C entre Arduinos

Per fer servir el bus I2C en Arduino, el IDE Standard de la marca proporciona la llibreria "Wire.h", que conté les funcions necessàries per controlar el hardware integrat però això ho explicarem més endavant en el desenvolupament software.

4.2 Software

En aquest apartat pretenc descriure les funcions, llibreries i sentències que he realitzat en els diferents llenguatges de programació per tal d'implementar el projecte proposat. Seguirem els passos de l'estructura de software començant per Processing i acabant amb Python, amb la implementació dels scripts d'Arduino a la meitat.

4.2.1 Processing

4.2.1.1 Descripció i estudi previ

El primer punt abans de començar a programar codi és estudiar quines funcions haurà d'executar el nostre programa de Processing. El projecte en aquest apartat del software haurà d'ocupar-se de tres factors diferenciats.

El primer és el que s'encarregarà d'agafar l'àudio provinent del micròfon, processar-lo i enviar-lo al port serial. Aquest apartat és el més complicat i el que generarà més consum al programa.

El segon apartat que s'haurà d'ocupar serà la interactuació del programa amb l'usuari en temps real. Això es tradueix en que l'usuari pugui canviar el tipus de visualització amb diferents programes i també donar-li l'oportunitat de canviar de color.

El tercer i últim apartat és el que s'haurà d'encarregar de que el programa generi un entorn visual còmode i comprensible per l'usuari.

4.2.1.2 Captació i processament d'àudio

Aquest és el començament de tot el projecte, la captació i el processament d'àudio en temps real esdevindrà el primer pas per començar a treballar. Primerament haurem de saber quina llibreria importar a aquest script gràcies a la qual podrem treballar amb el so exterior en temps real. La llibreria és:

```
import processing.sound.*;
```

Aquesta llibreria ens proporcionarà les classes necessàries tant per obtenir el so com per transformar aquest so en un tipus de dada manejable. Un cop

descarregada i instal·lada la llibreria oficial de so des de la web de Processing podem treballar amb ella. El següent pas serà declara dues variables globals que ens marcaran les bases;

```
AudioIn input;
```
// Aquesta classe es declara per definir l'entrada d'àudio del nostre PC i poder treballar amb ella.

```
Amplitude rms;
```
// Aquesta classe es declara per definir una variable del tipus amplitud de la senyal d'àudio que entri per la targeta de so, en aquest cas el micròfon.

```
int scale=1;//
```
Aquesta variable del tipus numèrica enter ens servirà per igualar-la amb la amplitud generada pel so i treballar amb ella.

Ambdues declaracions de variables es faran abans de les funcions principals de **setup()** i de **draw().**

El proper pas en aquest apartat serà definir la disposició de tots els elements necessaris per treballar amb aquetes classes. Això es farà en la funció principal de *setup*() on posarem les condicions del programa.

La primera de les condicions serà igualar la variable de la classe del tipus àudio in amb una nova entrada d'àudio:

```
input = new AudioIn(this, 0);
```
// Aquesta sentència iguala la classe que s'ha declarat anteriorment amb una entrada d'àudio nova fent servir la llibreria *AudioIn*. El *this* especifica la referència al programa actual i el 0 el número de l'entrada d'àudio. En aquest cas només es disposa d'una entrada, la del micròfon, per això és la 0. En el cas de que existissin més entrades instal·lant una targeta de so externa es podria seleccionar la més adient.

La segona sentència que haurem d'escriure serà l'encarregada d'inicialitzar aquesta entrada d'àudio:

```
input.start(); // Aquesta funció és pròpia de la classe AudioIn. Inicia la entrada.
```

A continuació s'hauran de posar les bases per començar a treballar amb els nivells d'amplitud que entrin per l'entrada d'àudio prèviament generada. L'estructura a seguir és molt semblant a l'anteriorment descrita. El primer pas serà igualar la variable de la classe amb la seva inicialització:

```
rms = new Amplitude(this); //Aquesta frase iguala la variable amb un tipus de la seva classe per començar a treballar.
```

Un cop ja igualada la variable amb un tipus de la seva classe haurem de inicialitzar la monitorització de l'amplitud:

```
rms.input(input);// Amb aquesta declaració li diem al programa que agafi l'input declarat per inicialitzar el mètode de captació de la classe.
```

Un cop definides aquestes sentències ja ens podem situar a la funció principal de **draw()** per començar a treballar amb el procés d'àudio en temps real. El primer que farem en el programa principal serà convertir el valor retornat per la classe Amplitude en un numèric del tipus enter. Això ho farem declarant el següent:

```
scale=int(map(rms.analyze(), 0, 0.5, 1, 350)); // La funció map transforma un número definit en un rang el quantifica en un rang diferent i més còmode per treballar. El primer valor és el que es convertirà i els següents 4 són els rangs que s'han d'intercanviar.
Posteriorment la funció int agafarà aquest nou rang generat i el convertirà en un enter per poder treballar amb ell.
```

A partir d'aquest moment ja tenim un valor numèric del tipus enter el qual s'anirà actualitzant en temps real amb el nivell d'amplitud del so que arriba pel micròfon. Ara ja podem dissenyar un algoritme que treballi amb aquets valors. Es declararan dues funcions per treballar amb aquesta informació: La primera serà un algoritme que a partir de cert nivell de so o llindar, enviï una senyal d'activació, això ens serà molt útil per començar a dibuixar figures cada cop que existeixi senyal sonora. En la segona funció interpretarem quin nivell de senyal arriba fen diferents blocs d'amplitud que ens servirà per dibuixar un vúmetre.

El primer cas utilitzarem la funció *if/else* per delimitar el llindar de la variable *scale*:

```
if (scale >20) {
```
// Amb aquesta declaració posem com a llindar un nivell, el qual , a partir de 20 s'enviarà el senyal que acredita que hi ha so arribant al micròfon. Just després hi haurà un else perquè s'actualitzi en tot moment que no arriba senyal si baixa d'aquest llindar.

Pel segon algoritme utilitzarem la funció if, però amb la seva variant *else if* que ens donarà les eines per fer blocs d'amplitud:

```
if (scale > 0 && scale<=10){
```
// Aquesta sentència declarà el primer bloc en el rang 0 a 10 de senyal.

```
else if (scale > 10 && scale<=19){
```
// Aquesta sentència és el següent bloc i això s'anirà repetint amb els següents rangs:

- `scale > 19 && scale<=27`
- `scale > 27 && scale<=33`
- `scale > 33 && scale<=38`
- `scale > 38 && scale<=42`
- `scale > 45 && scale<=45`
- `scale > 45 [últim rang]`

Depenent de en quin rang se situí el valor actual de la variable *scale*, s'enviarà un caràcter o un altre al port serial perquè sigui interpretat per l'Arduino.

4.2.1.3 Encapsulament i enviament al port sèrie

Un cop processat l'àudio en temps real, s'ha d'enviar el missatge correcte al port sèrie per que aquest arribi a Arduino. En el port sèrie enviarem 4 caràcters del tipus ASCII. Per la senyal corresponent al processat d'àudio ens interessaran el segon i el tercer caràcter. El grup de caràcters que enviaran seran:

- **2on caràcter:** [A:B] --> on/off del senyal
- **3er caràcter:[**S,T,U,V,W,X,Y,Z]--> 8 nivells de volum

Per enviar el missatge s'haurà de descarregar i instal·lar la llibreria del port sèrie pròpia de Processing. Amb aquesta comanda al principi del nostre script podrem treballar amb la llibreria:

```
import processing.serial.*; // Declarem la llibreria
```

Un cop declarada podem començar a programar amb les eines que ens ha proporcionat la mencionada llibreria. Primer s'han d'inicialitzar algunes variables globals.

```
Serial myPort; // Aquesta sentència declara una variable del tipus
```

serial amb els mètodes necessàries per establir les connexions amb el port serial.

```
char a;
char b;
char c;
char  d; // Aquestes variables del tipus caràcter les utilitzarem per guardar els caràcters temporals que es van generant en temps real.

char data[ ] = {'1', 'A', 'S','A'}; // per últim declararem una cadena de caràcters amb 4 posicions que és el missatge final que s'enviarà al port sèrie després de cada iteració. Ens serà molt útil per enviar missatges en blocs i no perdre cap informació en cap moment.
```

Un cop tenim declarades les variables que entraran al joc del nostre script, hem d'escriure les condiciones que necessitem a la funció principal de *Setup*() per que el nostre programa funcioni correctament. Aquestes són molt senzilles, la primera és extreure el nom del port i la segona inicialitzar la variable de la classe serial.

```
String portName = Serial.list()[1]; // Amb aquesta sentència es pot extreure directament el nom del port sèrie utilitzat per l'Arduino. En cada sistema operatiu es pot dir de diferents maneres. Aquesta funció extrau de la memòria interna quin és el nom del port i el guarda a la variable del tipus String.  Això fa que la funció pugui funcionar en diverses plataformes.

myPort = new Serial(this, portName, 9600); // Amb aquesta declaració inicialitzem el port sèrie a la variable prèviament declarada, amb el nom exacte del port i amb la velocitat de transmissió per defecte: 9600.
```

Ara ja tenim les condicions necessàries per implementar el nostre programa i enviar el missatge al port serial. En la funció draw() el missatge passarà per tres fases:

La primera és el recull de la informació per mitja dels caràcters, en el cas del procediment d'àudio actuaran les variables "b" i "c". Si el llindar és mes gran que 20 com s'ha explicat anteriorment, la variable "b" obtindrà un valor 'A'. Si pel contrari el llindar baixa de 20, el valor de la variable serà 'B'. Cada cop que s'avaluï aquesta condició la variable es guardarà a la segona posició de la cadena declarada:

```
data[1]=b; // La variable "b" es guarda a la segona posició de la cadena.
```

L'avaluació dels nivells de volum segueix la mateixa pauta descrita abans i depèn de quin nivell arribi el so prendrà valors entre aquests: S,T,U,V,W,X,Y,Z.
On S és el nivell més baix i Z el més alt. Després es guardarà a la tercera posició de la cadena de caràcters:

```
data[2]=c; // La variable "c" es guarda a la tercera posició de la cadena.
```

Ara ja tenim implementat l'algoritme que avaluï en temps real el processament d'àudio i l'escrigui de forma ordenada i controlada a una variable. Ja podem procedir a enviar el missatge al port sèrie, però abans ens faltarà una última conversió. Per tema d'interpretació entre codis, transformarem la nostre cadena de caràcters (*arrray*) en una variable del tipus *string*.

```
String str2 = new String(data); // Aquesta conversió convertirà
```
la cadena de caràcters a una variable temporal del tipus string que es construirà en cada iteració.

Ara ja tenim una variable robusta que entra sencera al port i que l'Arduino interpretarà perfectament per enviar-la a la Raspberry. Ara només queda omplir el port sèrie amb aquest missatge, això es farà de la següent forma:

`myPort.write(str2);` // El mètode *write* és propi de la classe serial i escriu el missatge de tipus string.

Un cop arribat aquí, ja està enviat el missatge al port serial amb la informació actualitzada en temps real.

4.2.1.4 Interacció amb l'usuari

Per realitzar aquesta funció ens servirem d'alguns recursos ja utilitzats al processament d'àudio i de l'enviament al port sèrie. Aquest apartat haurà de ser capaç de rebre informacions de les preferències de l'usuari per enviar-les a la placa Raspberry i que el programa de visualització canviï o modificar el color.

Per realitzar la comunicació entre l'usuari i l'aplicació farem servir el teclat del nostre PC/MAC. Així ens servirà qualsevol tipus de hardware per executar el programa ja que aquesta funció està a tots els dispositius.

Es pot aprofitar una funció interna de Processing que avalua si alguna tecla ha estat premuda. Aquesta funció no necessita la declaració de cap llibreria, ni tampoc establir cap condició inicial en la funció *Setup*().

El seu funcionament és molt senzill; Es tracta d'una condició del tipus IF, on passem per paràmetres la paraula clau keyPressed i la igualem a *true*:

```
if (keyPressed == true)
```

Un cop ha entrat a dins del IF he generat una funció que avaluí quina tecla s'ha pres. El valor de la tecla, és a dir quina tecla exactament s'ha premut es guarda a una variable interna anomenada key . Dins de la nostre funció només hi han dos rangs diferenciats que són vàlids com a tecles:

- Del ‘1’ al ‘5’ per programes.
- De la ‘q’ a la ‘t’ per colors.

Dins de la “`void miFuncion()`” avaluarem amb dos mètodes del tipus *if-else if* si és d’un rang o de l’altre. Si és del tipus programa es guardarà a la variable *char* ‘a’ que hem declarat a l’anterior fase(enviament serial), valors que van de l’1 al 5 en versió caràcter.
Si és del tipus color la tecla que s’ha seleccionat, es desarà a la variable *char* ‘d’, valors que van des de ‘A’ fins a ‘E’.
Un cop desat el valor adient a la variable, es disposarà a posar aquests valors dins de la cadena de caràcters data, a la posició 1 i 4 respectivament:

- `data[0]=a;`
- `data[3]=d;`

Un cop desada se seguiran els mateixos passos descrits anteriorment per enviar la informació al port sèrie.

4.2.1.5 Entorn visual per l’usuari

Aquest apartat es descriuran els passos que s’han de seguir per desenvolupar un entorn visual amable i clar per l’usuari.

Primer s’ha de declarar el pop-up de la mida que es vulgui i els colors que es desitgin:

> `size(640,360);//` Aquestes sentencies en primer lloc declaren la mida en numero de *pixels* (X,Y) que es desitja que tingui l’aplicació. Aquesta declaració s’ha d’ubicar dins de la funció principal *Void Setup*().

Un cop tenim declarada la mida de l'aplicació es poden definir els colors de fons i de text. Aquestes sentencies s'han de posar dins de la funció *Void Draw*():

`background(125,255,125);` //Aquesta especifica quin color té el fons d'aquesta aplicació. En aquest cas verd.

`fill(255,0,150);` // Aquesta sentencia posar el color de text. En aquest cas violeta.

Ara ja es pot començar a escriure les ordres per construir l'entorn d'usuari. Per fer això s'utilitzarà la classe pròpia de Processing Text. Aquesta no necessita importació de ninguna llibreria. Dins de la funció *Draw*() es seguirà la següent estructura per escriure per pantalla:

`text("  text per pantalla  ", X, Y);` // Aquesta sentencia de la classe text, només necessita 3 paràmetres. El primer es el text que es desitja mostrar a la pantalla. El següent és la posició X d'on es situarà la caixa amb el text i la posició Y es lo mateix però per el eix Y.

El Interface de l'aplicació mostrarà com la figura 16:

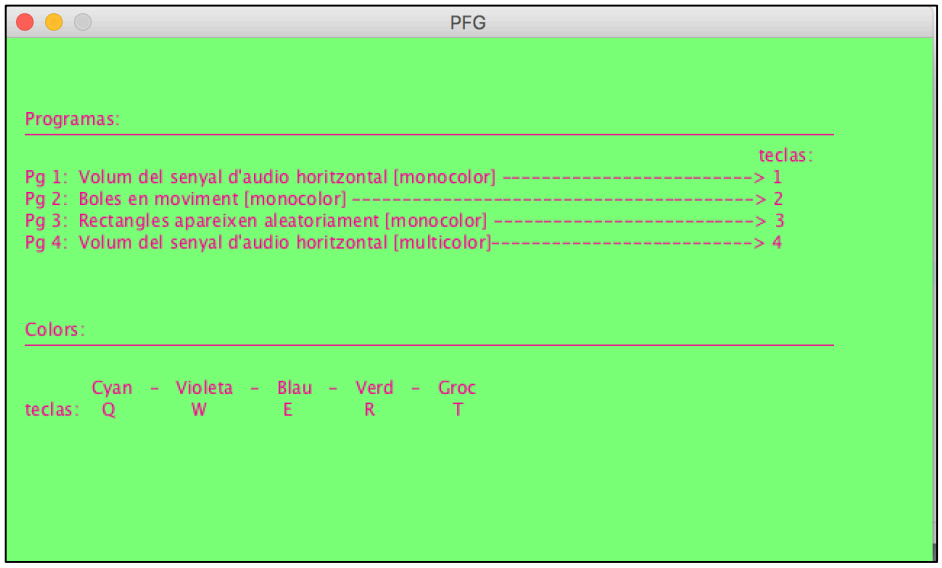


Figura 16. Pantalla del programa principal

Aquí s'aconsegueix un entorn amable i clar perquè l'usuari pugui fer servir l'aplicació sense tenir coneixement previs d'enginyeria .

4.2.2 Arduino IDE

4.2.2.1 Descripció i estudi previ

L'Arduino és el punt que connecta el PC amb la Raspberry, per tant és l'encarregat de generar, executar i controlar el protocol de comunicació, tant sèrie com principalment bus I2C.

La seva feina es dividirà en dues fases; L'enviament de dades al bus I2C i l'altre la recepció de les mateixes al bus I2C.Per establir la connexió al bus I2C utilitzarem la llibreria "Wire.h"

4.2.2.2 Enviament de dades al bus I2C

Aquest fase de l'Arduino consta de diverses parts. Bàsicament la placa encarregada de la transmissió haurà de: Rebre la informació des del port serial provinent de Processing, processar després aquest missatge i per últim enviar-lo amb la codificació correcte per la recepció.

4.2.2.2.1 Recepció del port Serial

Per rebre informació o enviar-la a traves del port serial no necessitem declarar cap llibreria especial ja que Arduino està preparat per interactuar amb aquest

port d'informació. El que si que haurem de fer és declarar unes condicions inicials a la funció principal de Setup() del nostre script.

> `Serial.begin(9600);` // Afegint aquesta sentència estem dient al port serial que s'inicialitzi per rebre o enviar informació.

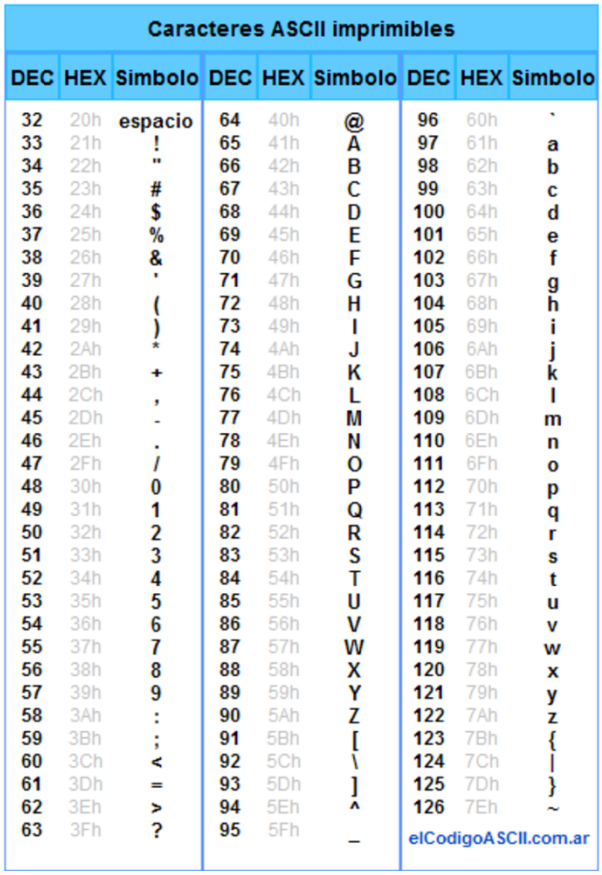
Un cop inicialitzat, ja podem començar a treballar amb el port dins de la funció principal Loop(). Per treballar només quan arribi informació (realment és tota l'estona) utilitzarem un bucle:

> `while (Serial.available()) {` // Entrarem a aquest bucle cada cop que el port estigui disponible i preparat per la connexió. És molt important declarar aquesta condició ja que sinó no obtindríem correctament i per ordre els caràcters des de Processing.

Just després d'entrar al bucle, farem 4 lectures individuals i concatenades per extreure la informació de cada caràcter de forma ordenada:

```
int val = Serial.read();
int val2 = Serial.read();
int val3 = Serial.read();
int val4 = Serial.read();
```

Amb aquestes quatre sentències anirem llegint els caràcters de 4 en 4 i posats de forma individual a les 4 variables del tipus enter; val,val2,val3,val4. La funció que retorna el valor que hi ha al port serial (`Serial.read()`) s'iguala a un enter, ja que encara que en Processing s'ha enviat un caràcter la funció retorna al seu valor decimal de la taula ASCII (figura 17):



Caracteres ASCII imprimibles								
DEC	HEX	Simbolo	DEC	HEX	Simbolo	DEC	HEX	Simbolo
32	20h	espacio	64	40h	@	96	60h	`
33	21h	!	65	41h	A	97	61h	a
34	22h	"	66	42h	B	98	62h	b
35	23h	#	67	43h	C	99	63h	c
36	24h	$	68	44h	D	100	64h	d
37	25h	%	69	45h	E	101	65h	e
38	26h	&	70	46h	F	102	66h	f
39	27h	'	71	47h	G	103	67h	g
40	28h	(	72	48h	H	104	68h	h
41	29h	)	73	49h	I	105	69h	i
42	2Ah	*	74	4Ah	J	106	6Ah	j
43	2Bh	+	75	4Bh	K	107	6Bh	k
44	2Ch	,	76	4Ch	L	108	6Ch	l
45	2Dh	-	77	4Dh	M	109	6Dh	m
46	2Eh	.	78	4Eh	N	110	6Eh	n
47	2Fh	/	79	4Fh	O	111	6Fh	o
48	30h	0	80	50h	P	112	70h	p
49	31h	1	81	51h	Q	113	71h	q
50	32h	2	82	52h	R	114	72h	r
51	33h	3	83	53h	S	115	73h	s
52	34h	4	84	54h	T	116	74h	t
53	35h	5	85	55h	U	117	75h	u
54	36h	6	86	56h	V	118	76h	v
55	37h	7	87	57h	W	119	77h	w
56	38h	8	88	58h	X	120	78h	x
57	39h	9	89	59h	Y	121	79h	y
58	3Ah	:	90	5Ah	Z	122	7Ah	z
59	3Bh	;	91	5Bh	[	123	7Bh	{
60	3Ch	<	92	5Ch	\	124	7Ch	|
61	3Dh	=	93	5Dh	]	125	7Dh	}
62	3Eh	>	94	5Eh	^	126	7Eh	~
63	3Fh	?	95	5Fh	_	elCodigoASCII.com.ar		

Figura 4. Taula de caràcters ASCII

És a dir, per a la interpretació dels programes que es guarden a la variable **val**, la traducció quedaria de la següent manera figura 18:

Programa	Caràcter enviat des de Processing	Valor de l'enter rebut a Arduino
1	'1'	49
2	'2'	50
3	'3'	51
4	'4'	52
5	'5'	53

Figura 18. Taula de valors

Mentre que la taula de colors quedaria traduïda de la següent manera i guardada a la variable **val4(**Figura 19):

Color	Caràcter enviat des de Processing	Valor de l'enter rebut a Arduino
1	'A'	65
2	'B'	66
3	'C'	67
4	'D'	68
5	'E'	69

Figura 19. Taula de valors de variable Val4.

Els 8 nivells de volum estaran guardats a la variable **val3** (figura 20), quedarien:

Nivells de volum (de més baix a més alt)	Caràcter enviat des de Processing	Valor de l'enter rebut a Arduino
1	'S'	83
2	'T'	84
3	'U'	85
4	'V'	86
5	'W'	87
6	'X'	88
7	'Y'	89
8	'Z'	90

Figura 20. Taula de valors de variable Val3

Per últim, quedarà el valor del llindar, que tindrà dues possibles interpretacions. La variable **val2** obtindrà els valors d'aquesta taula:

Llindar	Caràcter enviat des de Processing	Valor de l'enter rebut a Arduino
El supera	'A'	65
No el supera	'B'	66

Figura 21. Taula de valors de variable Val2

Ara ja esta descrit com Arduino rep el missatge des del port serial i l'interpreta.

4.2.2.2.2 Enviament del missatge a través de I2C

En aquest apartat se centrarà en agafar el valors enters definits a l'apartat anterior i enviar-los d'una forma correcta al bus I2C perquè l'aparell receptor el pugui interpretar sense errades.
Primer s'ha de declarar la llibreria amb els mètodes necessaris per fer ús del bus I2C en Arduino. Aquesta llibreria és la Wire.h, la que ens permet realitzar aquesta connexió.

```
#include <Wire.h>
```
// Aquesta sentència al principi del programa ens permetrà treballar amb les eines de la llibreria.

Ara hem d'establir les condicions inicials a la funció *Setup*(), que inicialitzaran la connexió I2C:

```
Wire.begin();
```
// Aquest mètode inicialitza la llibreria i connecta l'Arduino al bus. Sinó s'especifica l'adreça (és deixa el parèntesis buit), Arduino es connectarà al bus com a *Master*. Per aquest projecte ja ens interessa que es connecti com a *Master* ja que aquesta placa només tindrà que enviar i mai rebre informació.

Un cop tenim la llibreria inicialitzada s'ha d'enviar el missatge codificat correctament. Això es farà mitjançant una funció *if-else if* molt llarga on en cada condició entraran varis paràmetres concatenats amb l'ordre lògic **AND**. Per exemple quant arribi des de Processing la següent instrucció:

- **Programa** : 1
- **Llindar** : El supera
- **Nivell de volum** : 1
- **Color** : 1

La condició que complirà serà la de :

```
if(val2==65 and val==49 and val3==83 and val4==65 ){
```

Un cop entrat a algun apartat de la funció de condicions, el procés d'enviament sempre segueix la mateixa estructura:

`Wire.beginTransmission(8);` // Aquest mètode inicia la transmissió de dades al dispositiu esclau. El paràmetre '8' indica la direcció de destí de la informació

`Wire.write("0FSA");` // El mètode *write* com el seu nom indica escriu la informació al bus a l'adreça prèviament indicada. El missatge està en format string i porta la informació de la instrucció correcte per aquesta condició.

`Wire.endTransmission();` // Aquesta funció finalitza la transmissió a l'esclau iniciada pel *beginTransmission*. Aquesta funció no retorna res.

4.2.2.3 4.2.2.3 Recepció de dades del bus I2C

Aquest apartat és molt més senzill que l'anterior ja que no processa el missatge que arriba des del bus I2C. Bàsicament agafa el missatge i el posa a disposició del port serial. Posteriorment l'script de Python serà l'encarregat de processar la informació que conté aquest missatge.

Llavors aquest fase del projecte es dividirà en dues parts; Agafar el missatge del bus i portar-lo al port serial.

4.2.2.3.1 Recepció del missatge des del bus I2C

Per fer aquest procés s'ha de declarar com abans la llibreria Wire.h per treballar amb el bus. També s'hauran d'especificar certes condicions inicials a la funció *Setup*(). Aquestes varien una mica de l'apartat anterior:

`Wire.begin(8);` // amb aquesta declaració inicialitzem la placa Arduino perquè realitzi connexions del tipus bus I2C. Com entre parèntesis està escrit el '8', vol dir que aquest dispositiu es convertirà en esclau amb la direcció 8.

`Wire.onReceive(receiveEvent);` // Registra una funció que serà cridada quan un dispositiu esclau rebi una transmissió des d'un mestre. Aquesta funció la farem servir perquè s'activi en tot moment la placa quant rebi qualsevol cosa des del mestre. Això ens farà tenir una connexió molt més fluida que en el moment de pintar la pantalla de leds es veurà molt millor.

Un cop declarades les condicions inicials de la recepció de dades des de l'Arduino mestre, s'ha de programar una funció perquè les rebi correctament i les desi. Aquesta funció estarà inicialitzada al Setup() i tindrà la condició de que fins que no es connecti l'esclau amb el mestre no començarà a guardar els valors.

`void receiveEvent(int howMany){` // Declaració d'una funció que s'executa cada cop que rebi un missatge des de l'Arduino mestre. El paràmetre *HowMany* especifica el número de bytes que rep al missatge.

`while (1 < Wire.available()){` //Aquesta condició estableix que mentre el bus I2C estigui disponible i retorni un número de bytes superior a 1 es mantingui l'execució de les comandes de dins del bucle.

`char c = Wire.read();` //Aquesta sentència inicia la lectura dins del bucle abans descrit. Posa cada caràcter definit per 1 byte dins de la variable 'c' del tipus char.

Quant només queda un caràcter dins del missatge rebut pel bus I2C surt del bucle. Just després del bucle fa l'última lectura de l'últim byte i el desa a una variable 'x' també del tipus char.

4.2.2.3.2 *Enviament del missatge al bus I2C*

En aquest apartat només ha d'agafar la informació continguda a les variables i enviar-la de forma correcta al port serial perquè l'script de Python la pugui aprofitar. Per fer això ens tornarem a aprofitar dels serveis que proporciona Arduino amb la seva classe serial.

Com ja s'ha descrit no s'ha d'importar la llibreria i s'ha de inicialitzar a comunicació via port-serial al *Setup*(). Un cop fet això farem servir el mateix mètode dues vegades però amb una petita variació.

`Serial.print(c);` // Aquesta funció estarà dins del bucle mentre el missatge ocupi més espai d'1 byte. Escriurà un caràcter darrera l'altre concatenats entre sí en una mateixa línia.

`Serial.println(x);` // Aquest mètode s'executarà just després de la lectura de l'últim byte contingut en el missatge. També escriu la variable del tipus char concatenat amb els altres caràcters. La diferència és que després d'això farà un salt de línia que ajudarà a que Python rebi el missatge per blocs i l'interpreti sense interferències.

Un cop arribat aquí ja està feta la transmissió del missatge des del bus I2C fins al port serial. Enviat en blocs correctament i controlat per la placa Arduino.

4.2.3 Python

4.2.3.1 Descripció i estudi previ

Aquest llenguatge de programació és l'encarregat d'executar les comandes per a la il·luminació de la pantalla de leds. Mitjançant llibreries dibuixarem figures amb diferents modes. Totes aquestes figures estaran controlades d'una forma o d'un altre pels missatges arribats des de Processing.

Aquest script estarà format per varies etapes, casi totes elles funcions, que executaran els programes.

4.2.3.2 Recepció de dades port serial en la funció principal

El primer pas és definir la funció principal. En aquest cas serà un bucle infinit que reproduirà la seqüència de lectura fins que es trenqui la condició. Es declararà una variable global 'x' al principi , que sigui del tipus numèric enter amb un valor inicialitzat de 1. Mes tard s'escriurà la sentència:

```
while x>0: // Amb aquesta declaració fem que el bucle sigui infinit fins
que la variable canvi a un valor més petit que 1.
```

Aquesta serà la nostre funció principal. Més tard importarem al principi de l'script la llibreria serial i inicialitzarem la classe.

```
import serial
arduino = serial.Serial('/dev/ttyACM0', baudrate=9600,
timeout=0.250)
```

En aquesta última sentència s'inicialitza la classe del tipus serial i es guarda a la variable Arduino. Els paràmetres es defineixen:

- `'/dev/ttyACM0':` Nom del port serial, així es diuen els ports a les màquines de UNIX.
- `baudrate=9600:` És la velocitat de transmissió del port, ha de coincidir amb l'executada en Arduino.
- `timeout=0.250:` És el temps que espera cada cop que llegeix. És un temps baix perquè no existeixi error de lectura, però suficientment gran per no sobre explotar la memòria de la placa.

Un cop tenim inicialitzada la classe, podem començar a treballar dins del bucle principal amb la lectura continuada de missatges al port serial. Això es farà amb:

`line = arduino.readline()` //Aquest mètode de la classe retorna la lectura dels missatges que van apareixent al port. Com prèviament el missatge s'ha tallat amb salts de lineal, la funció retornarà cada línia individualment. Aquesta cadena de caràcters es guardarà a una variable del tipus string anomenada *line*.

Ja tenim una línia rebuda des del port sèrie, el proper pas és desgranar-la en caràcters individuals. Recordem que cada caràcter porta amb ell la informació per dirigir l'aplicació, és a dir, és un pas molt important del projecte.

```
f = line[0:1]
f2 = line[1:2]
f3 = line[2:3]
f4 = line[3:4]
```

Aquesta, és la manera que té Python per accedir a punts dins d'una cadena de caràcters. Les variables **f,f2,f3 i f4** obtindran el valor del caràcter que està a la posició **0,1,2 i 3**, respectivament.

Per últim escriurem per la terminal el missatge que arriba cada moment a la variable *line*. Això serà molt útil per monitoritzar el senyal en el moment de crear les funcions que dibuixin els efectes.

4.2.3.3 Estructura del programa

En aquest apartat s'explicarà quin criteris s'han seguit per estructurar el programa. Bàsicament, hi ha una funció que examina quin programa s'està demanant i quin color esta predeterminat.

Aquesta funció serà cridada just després de la recepció del missatge per part del port serial.

```
def func1():
        if f == "0" and f2 == "F":
               func4()
        if f == "0" and f2 == "E":
          func5()
        if  f == "0"  and f2 == "D":
          func3()
        if f == "0" and f2 == "G":
          func2()
        if f == "C" :
               print(f)
               del draw
               matrix.Clear()
        line = arduino.readline()
        f = line[0:1]
        f2 = line[1:2]
        f3 = line[2:3]
        f4 = line[3:4]
```

El que es mostra a la funció func1(), com estructura l'script la manera de processar. El primer que s'observa és el conjunt de condicions que determinaran quin programa s'escull entre els possibles. El tipus de condició necessita de dos paràmetres; **f** i **f2.**

La variables **f** ens proporciona la certesa que hi ha àudio entrant pel micròfon. Això marca que sempre es reproduirà la imatge que sigui en cada programa si hi ha musica, és a dir la imatge es dibuixarà al ritme de la música.

La condició amb la variable **f2**, ens donarà informació sobre quin és el programa que l'usuari ha desitjat que es dibuixi. Aquesta variable té guardada la informació relativa amb aquest sentit.

Un cop ha arribat hi ha missatge amb so i una tecla correcta de programa totes les funcions condició actuen de la mateixa manera. Criden a una altre funció que ha estat creada per dibuixar el programa seleccionat. Cada una serà diferent de l'altre i en algunes es podrà variar el color.
L'última condició és especial, només s'executarà quan el llindar del so sigui més baix a l'indicat, és a dir f serà igual al caràcter "C". Això és així per assegurar que no es queda cap programa dibuixant quan representa que no hi ha so. És una forma més, per estabilitzar la propietat d'imatge escrita al ritme de la música.

4.2.3.4 Programes

Aquí descriurem els passos necessaris per dibuixar figures a la pantalla de leds i com estan controlades des del programa principal de Processing. El primer que farem és estudiar i saber com declarar les llibreries que necessitem pel projecte.

4.2.3.4.1 Llibreries

Necessitarem varies llibreries de Python, algunes estan directament relacionades amb el hardware de la pantalla de leds, unes altres són interesants per jugar amb les figures.

`import Image` : Aquesta llibreria declara una imatge amb uns determinats píxels, cada píxel és correspost pels píxels reals del mòdul de leds.

`import ImageDraw` : Llibreria que té els mètodes per dibuixar. Per exemple línies, rectangles, rodones...

`import time` : Llibreria que conté la funció que retorna quantitat de temps. És molt útil per algun programa on les figures es tinguin que moure, poden controlar la velocitat imposant parades de cert temps entre figura i figura.

`from rgbmatrix import Adafruit_RGBmatrix` : Aquesta llibreria es pròpia de la marca que proporciona el hardware. S'ha de descarregar, instal·lar i declarar l'script.

`import random` : Llibreria pròpia de Python que conté els mètodes necessaris per generar números aleatoris.

`import threading`: Aquesta llibreria de Python té les classes necessàries per executar funcions en *background*, és a dir, funcions que es vagin executant paral·lelament amb les altres.

4.2.3.4.2 Declaració de variables

Aquí descriurem les variables globals mes importants que s'han de declarar al principi de l'script i per què s'utilitzen. Ens centrarem en 3 tipus de variables que són imprescindibles per dibuixar a la pantalla de leds, deixant de banda algunes variables que s'utilitzen però que són molt més comunes:

`matrix = Adafruit_RGBmatrix(16, 1)` // Aquesta sentència declara una variable que està inicialitzada amb la classe pròpia creada per Adafruit. Aquesta, té la funció de declarar la comunicació amb la pantalla de leds.

Els paràmetres que estan declarats són; en primer lloc el número de columnes que té la nostre matriu. En aquest cas particular 16. En segon lloc el número de blocs o matrius led que s'han necessitat per crear aquest número de columnes. És molt interesant aquest últim paràmetre ja que ens dona la possibilitat de muntar una matriu led tan gran com vulguem (tenint en compte el consum per bloc, cada bloc consumeix aproximadament 1.4 ampers).

`image = Image.new('RGB', (32, 32))` // Aquí s'està declarant una variable i s'està inicialitzant amb la classe *Image*. Aquest mètode crea un espai de dibuix que s'utilitzarà per posar les figures.

Els paràmetres que entren són dos també: El primer és l'espai de color en el que treballarem. És un espai del tipus RGB amb les barreges de colors pertinents, més endavant s'explicarà com és aquesta barreja. El segon paràmetres és una matriu de dues dimensions amb la mida de cada projecció; X i Y respectivament. En aquest cas serà de 32x32, que és més gran que la matriu que fem servir però serà útil per fer programes diferents.

`draw = ImageDraw.Draw(image)` //Aquesta variable també és un espai de treball però basat en el declarat anteriorment. Aquí es prepara la variable draw per dibuixar els mètodes continguts a la classe *ImageDraw*, les figures geomètriques.

L'únic paràmetre que s'ha de passar és el del entorn de dibuix declarat a la variable anterior.

4.2.3.4.3 Paleta de colors

Com s'ha declarat abans, l'espai de color en el que treballarem serà RGB. Per establir els colors de les figures realitzarem la barreja d'aquets colors primaris per aconseguir els que desitgem.

La fórmula per declarar els colors és molt senzilla; Es tracta de 6 caràcters en hexadecimal, dividits en tres blocs. El dos primers caràcters són els corresponents a la quantitat de color vermell, els dos següents són els corresponents al color verd i per últim hi haurà els corresponents al color blau.

Cadascun dels caràcters podran escriure des del 0 fins a la F , sent 0 el nivell mes baix i F el nivell mes alt. Per exemple el color vermell pur serà : "FF0000", ja que el nivell de vermell està al màxim i els de verd i blau al 0. El verd pur serà "00FF00" i el blau "0000FF".

La guia de colors es pot seguir amb la següent taula (figura 22):

2	Color 0:		[Color 0]	#FFFFFF	255	255	255	16777215
3	Color 1:		[Color 1]	#000000	0	0	0	0
4	Color 2:			#FFFFFF	255	255	255	16777215
5	Color 3:		[Color 3]	#FF0000	255	0	0	255
6	Color 4:		[Color 4]	#00FF00	0	255	0	65280
7	Color 5:		[Color 5]	#0000FF	0	0	255	16711680
8	Color 6:		[Color 6]	#FFFF00	255	255	0	65535
9	Color 7:		[Color 7]	#FF00FF	255	0	255	16711935
10	Color 8:		[Color 8]	#00FFFF	0	255	255	16776960
11	Color 9:		[Color 9]	#800000	128	0	0	128
12	Color 10:		[Color 10]	#008000	0	128	0	32768
13	Color 11:		[Color 11]	#000080	0	0	128	8388608
14	Color 12:		[Color 12]	#808000	128	128	0	32896
15	Color 13:		[Color 13]	#800080	128	0	128	8388736
16	Color 14:		[Color 14]	#008080	0	128	128	8421376
17	Color 15:		[Color 15]	#C0C0C0	192	192	192	12632256

Figura 22. Paleta de colors amb el seu codi hexadecimal i mescla.

4.2.3.4.4 Programa 1

Aquest programa serà el que estarà localitzat a la tecla amb el número 1 del PC amb Processing. Quan es premi aquest botó s'haurà d'executar aquesta funció únicament. El programa bàsicament és una visualització del nivell de volum amb diferents colors i que està orientat horitzontalment.

El primer que s'ha de saber és on està el punt inicial(0,0). Aquest punt es troba a l'esquerra (figura 23) de tot si tenim la matriu en horitzontal però en comptes de la part de sota el punt 0 de l'eix de les Y estarà a la part de dalt.

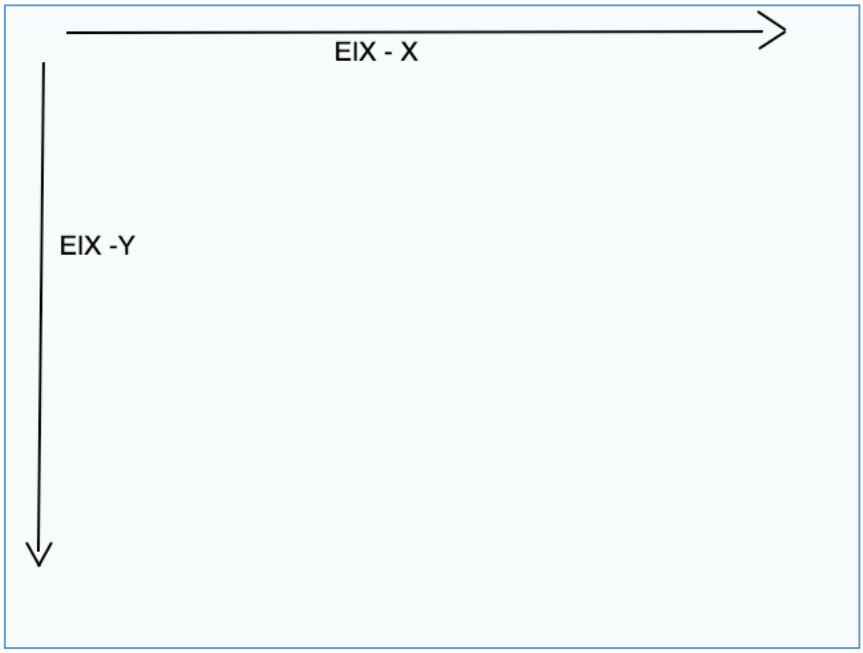


Figura 23. Estructura de la matriu led

Aquest factor fa que es tingui que pensar bé la declaració de les posicions dels objectes. El primer punt de la funció és un conjunt de condicions del tipus IF-ELIF que avaluen de quin color s'ha de pintar la figura . Depenent d'un o d'un altre entrarà dins d'una condició. Aquesta informació l'extraurà de la variable **f4.**

Si per exemple, el valor que indica variable és una ‘A’ entrarà a la primera condició.

```
draw3.rectangle((0, 0, 16, 16), fill="#fd00ff")
draw3.rectangle((16, 0, 32, 16), fill="#fd00ff")
```

Aquestes sentències utilitzen el mètode rectangle de la llibreria draw per la qual es pot dibuixar un rectangle amb la mida que es desitja i també especificant el color. El primer paràmetre s’encarrega de dibuixar el rectangle des d’un punt(X,Y) fins a un altre punt (X,Y). El segon paràmetre especifica de quin color es dibuixarà el rectangle seguint la pauta d’espai de color RGB hexadecimal.

Aquestes dues declaracions, dibuixen dos quadrats que ocuparien tota la matriu; ja que un aniria des de la X=0 fins a la X=16 ocupant tota l’alçada de la matriu, i l’altre des de X= 16 fins a X=32 .

Un cop tenim ja dibuixades les figures, entren en joc els nivell de volum que posaran aquets quadrats en una posició o una altre. Això es realitzarà amb un altre conjunt de condicions IF-ELIF. Els valors que avaluarà són quin caràcter hi ha a la variable **f4,** aquests caràcters aniran des de la ‘S’ fins la ‘Z’. La S correspon al valor més baix de volum i la Z el més alt.

```
matrix.Clear()
matrix.SetImage(image3.im.id,0, 15)
```

// Aquestes sentències són les encarregades de posicionar la imatge dibuixada amb els mètodes de la classe Draw. Primer es netejarà la matriu suprimint tota imatge present, és molt útil per evitar-nos leds que estiguin dibuixats prèviament d’un altre programa.

Just després es dibuixarà la imatge. El mètode *SetImage* agafa les figures dibuixades a la imatge que té com a id el primer paràmetre que li entra, en aquest cas els dos quadrats que s'han declarat més amunt.

Els dos següents paràmetres són la posició (X,Y) que prendrà aquesta imatge. Sent X i Y el nou centre de coordenades de la imatge (la seva posició 0,0) respecte a la matriu. En aquest cas com és el nivell de volum més baix, la imatge estarà quasi al nivell més baix i només dibuixarà una línia a la matriu led.

Aquest posició anirà variant fins arribar a cobrir tota la pantalla de leds amb la imatge dels dos quadrats concatenats quan el volum de la música sigui el més alt.

4.2.3.4.5 Programa 2

Aquest programa es reproduirà quan l'usuari premi la tecla 2 del teclat del PC. Dibuixarà uns cercles que començaran en un punt aleatori del eix de les X i en un punt extrem del eix de les Y. Seguidament es mourà amb una certa velocitat i en línia recta . Aquests cercles es dibuixaran cada cop que soni música.

Per programar aquesta visualització farem ús d'una funció especial de Python. Utilitzarem un objecte anomenat *Thread*. Aquest objecte té els mètodes necessaris per executar un procés mentre el programa principal segueix funcionant. Primer definirem la funció:

```
def worker():
        matrix.Clear()
          draw.arc((0,0,          4,4),start=0,end=360,
          fill="#fd00ff" )
        posX= random.randrange(0,28)
        i = 0
        for i in range (20):
            matrix.SetImage(image.im.id, posX, 16-i)
            time.sleep(0.025)
```

Primerament definim el nom de la funció "*worker*", un cop dins, el primer que s'ha de fer és netejar la matriu de leds amb la funció "clear". Seguidament declararem la figura que es dibuixarà en aquest programa; en aquest cas és un cercle de mida 4x4 píxels. Els paràmetres *start* i *end* defineixen la mida en graus que tindrà la circumferència: En aquest cas és un cercle complet. Per últim s'especifica el color RGB.

El següent pas és definir en quin punt del eix X es dibuixarà el cercle. La variable posX rebrà un punt aleatori des del 0 fins el 28 del eix X.

Un cop tenim tota aquesta informació, el següent pas és crear una funció tipus bucle que mogui la imatge del cercle per tota la matriu. Això es farà amb la funció for on la variable que s'avalua va des del 0 fins el 20. Quan es posa la imatge aquesta rebrà la posició aleatòria del eix de les X i sempre s'inicialitzarà a la posició 16-i en el eix de la Y, on i és una variable on sempre és 0 al principi del bucle.

Cada cop que s'incrementi un cicle del bucle la figura s'anirà movent una posició en el eix de les Y. La variable i tindrà com a màxim el número 20, ja que s'ha de contar amb l'altura de 4 píxels del cercle, que sumats als 16 píxels d'altura del mòdul led farà que desaparegui la figura en la posició = 16-20.

Per últim es declara una pausa de temps per fer el moviment més harmònic, en aquest cas la pausa entre moviment serà de 25 mili segons.

Un cop definida i explicada la funció de la classe *Thread*, es pot declarar la funció que cridarà a aquest últim. Cada cop que es cridi la funció *worker*() aquesta anirà dibuixant els cercles en moviment mentre el programa es segueix executant. La funció principal d'aquest programa serà:

```
def func5():
        t = threading.Thread(target=worker)
        t.start()
```

Aquesta funció és molt curta i es limita a declarar i cridar la funció de la classe *Thread*. Primer s'ha d'igualar una variable amb la classe *Thread* per donar-li els seus mètodes. Se li ha de passar per paràmetre el nom de la funció que haurà d'executar. Un cop definit això s'executarà el mètode d'inici del *Thread*, que reproduirà en segon pla la funció definida en el paràmetre.

Bàsicament el que fa la funció és crear un cercle i moure'l cada cop que el so superi el llindar especificat en Processing.

4.2.3.4.6 Programa 3

Aquest programa dibuixarà rectangles de 4 píxels d'amplada i 16 d'altura que es podran canviar de color i que apareixeran en una posició aleatòria del eix X cada cop que soni la música.

Per realitzar això disposarem d'una única funció que es dividirà en dos blocs:

```
    if f4 =='A':
            draw4.rectangle((0, 0, 4, 16),
fill="#00ffff", outline="#00ffff")
        elif f4 == 'B':
            draw4.rectangle((0, 0, 4, 16),
fill="#fd00ff", outline="#fd00ff")
        elif f4 == 'C':
            draw4.rectangle((0, 0, 4, 16),
fill="#5af60a", outline="#5af60a")
        elif f4 == 'D':
            draw4.rectangle((0, 0, 4, 16),
fill="#0000ff", outline="#0000ff")
        elif f4 == 'E':
            draw4.rectangle((0, 0, 4, 16),
fill="#f0ff00", outline="#f0ff00")
```

El primer bloc és l'encarregat de declarar la figura del rectangle i establir el color. Aquest conjunt de condicions s'encarrega d'establir el color del rectangle en funció del valor de la variable **f4.**

Un cop establer el color i declarada la figura, es podrà dibuixar la figura:

```
posX= random.randrange(0,28,4)
matrix.SetImage(image4.im.id, posX, 0)
```

Amb aquestes sentències farem un procés similar al programa 2. Primer es declara una variable on es guarda la posició X del rectangle a dibuixar. Aquesta posició serà un valor aleatori entre 0 i 28, amb la peculiaritat de fer salts de 4 posicions, és a dir, el valor de la variable serà 0,4,8... 28. Això ajudarà a fer el programa més maco visualment.

Un cop tenim la posició X, la següent sentència dibuixa aquest rectangle a la posició que s'ha generat.

4.2.3.4.7 Programa 4

En aquest apartat el programa farà la funció de visualitzar el volum de la música com el programa 1 però amb uns petits matisos. La primera diferència és que aquesta imatge se centrarà en imatges horitzontals i la segona que tindrà figures de diferents colors en el mateix espai visual.

Com els altres programes anteriors es basarà en una funció de condicions on la variable que jugarà el paper clau serà la variable **f3.** Aquesta variable és la que conté la informació corresponent amb nivell de volum que té el so que entra per la targeta de so el PC.

Des del PC arribaran 8 possible nivells de volum, és a dir existiran 8 condicions on es dibuixaran diferents figures amb diferents colors. El criteri per col·locar els colors a les figures és el típic en els leds que controlen el volum. Una part àmplia de color verd, una curta de color groc i per últim un extrem de color vermell indicant el volum més alt. Per aquest últim cas s'executarà la següent condició:

```
elif f3 == 'z':
     matrix.Clear()
     draw21.rectangle((0, 0, 4, 8), fill="#ff0000",
     outline="#ff0000")
     draw22.rectangle((0, 0, 4, 8), fill="#ff0000",
     outline="#ff0000")
     draw23.rectangle((0, 0, 4, 8), fill="#ff0000",
     outline="#ff0000")
     draw24.rectangle((0, 0, 4, 8), fill="#ff0000",
     outline="#ff0000")
     draw25.rectangle((0, 0, 4, 8), fill="#f0ff00",
     outline="#f0ff00")
     draw26.rectangle((0, 0, 4, 8), fill="#f0ff00",
     outline="#f0ff00")
     draw27.rectangle((0, 0, 4, 8), fill="#ff0000",
     outline="#ff0000")
      matrix.SetImage(image2.im.id, 0, 4)
      matrix.SetImage(image21.im.id, 4, 4)
      matrix.SetImage(image22.im.id, 8, 4)
      matrix.SetImage(image23.im.id, 12, 4)
      matrix.SetImage(image24.im.id, 16, 4)
     matrix.SetImage(image25.im.id, 20, 4)
     matrix.SetImage(image26.im.id, 24, 4)
     matrix.SetImage(image27.im.id, 28, 4)
```

En aquesta condició es pot observar com es declaren els rectangles que es dibuixaran. Hi han 7 rectangles declarats de 4x4 que corresponen als nivells de volum.(falta un nivell que està declarat just abans de les condicions)

Hi han 5 quadrats de color verd, 2 de color groc i 1 de color vermell. Just després d'això es declaren les sentències que col·locaran les figures a la pantalla. Aquestes figures es posaran concatenats a la meitat de la matriu.

4.2.3.5 Auto-execució del Script

En aquest punt s'exposaran els passos que s'han de seguir perquè el nostre script s'executi cada cop que la Raspberry s'inicia. Això és molt útil per dissenyar el prototip del producte ja que es podrà prescindir del teclat, del ratolí i de la pantalla. És a dir, el projecte final serà una caixa que només s'ha de connectar a la corrent i un cable tipus USB.

Aquest procés no és de Python sinó que és un propi del sistema operatiu Raspbian basat en Linux, però s'haurà de posar l'script Python a un lloc particular.

Només es necessita que el sistema operatiu Raspbian se li indiqui que ha d'executar l'script per l'usuari "pi". En una terminal escriurem el següent:

```
sudo nano /etc/profile
```

I al final del document incorporarem la següent línia:

```
sudo python /[path]/PFG.py
```

En el path escriurem la ruta que ha de seguir l'auto executor per trobar l'arxiu. PFG.py és el nostre script de Python amb totes les instruccions que s'han d'executar.

Amb aquets passos el nostre programa s'executarà cada cop que s'iniciï la Raspberry.

4.3 Disseny i fabricació del prototip

Per fer mes comercial el projecte, un pas molt important és l'encapsulament de tot l'equip hardware en un únic bloc, fen que l'usuari nomes hagi d'endollar la corrent i les dades perquè funcioni.

El primer que necessitem es saber les mides exactes dels components hardware per integrar-los en una caixa.

- Arduino UNO: *Length* – 68.7mm *Width* – 53.4mm
- Raspberry : *Lenght* – 85mm *Width* – 56mm

La distribució en la caixa serà la següent figura (24):

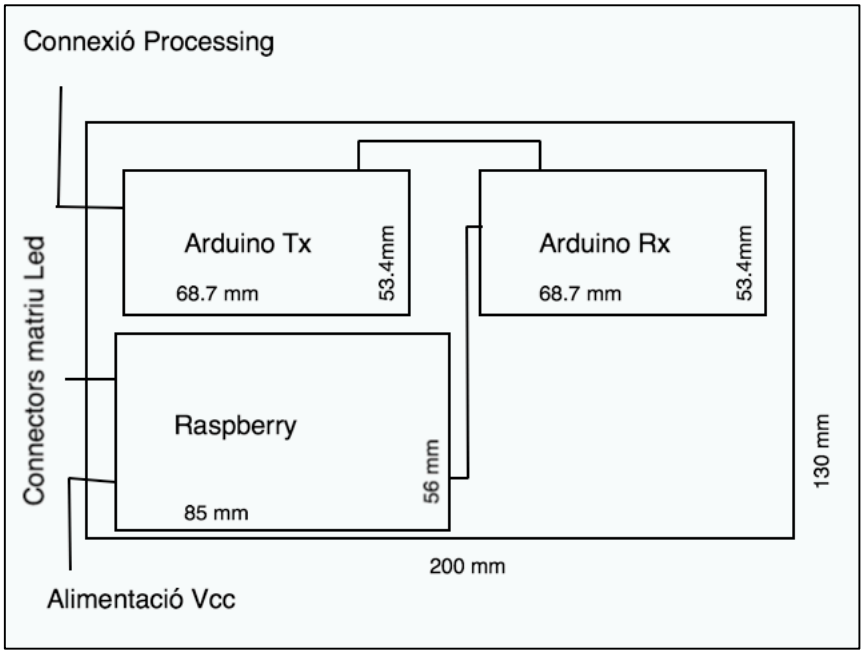


Figura 24. Esquema caixa prototip

La caixa del projecte serà una del tipus caixa de connexions elèctriques de 200x130x60 mm amb tapa de tornavisos. Per tal de poder desmuntar el mateix prototip per fer-li millores la Raspberry , a la caixa s'enganxarà una base especial de Raspberry que farà que quedi fixe però sense enganxar-la físicament.

El prototip final quedaria interiorment (figura 25):

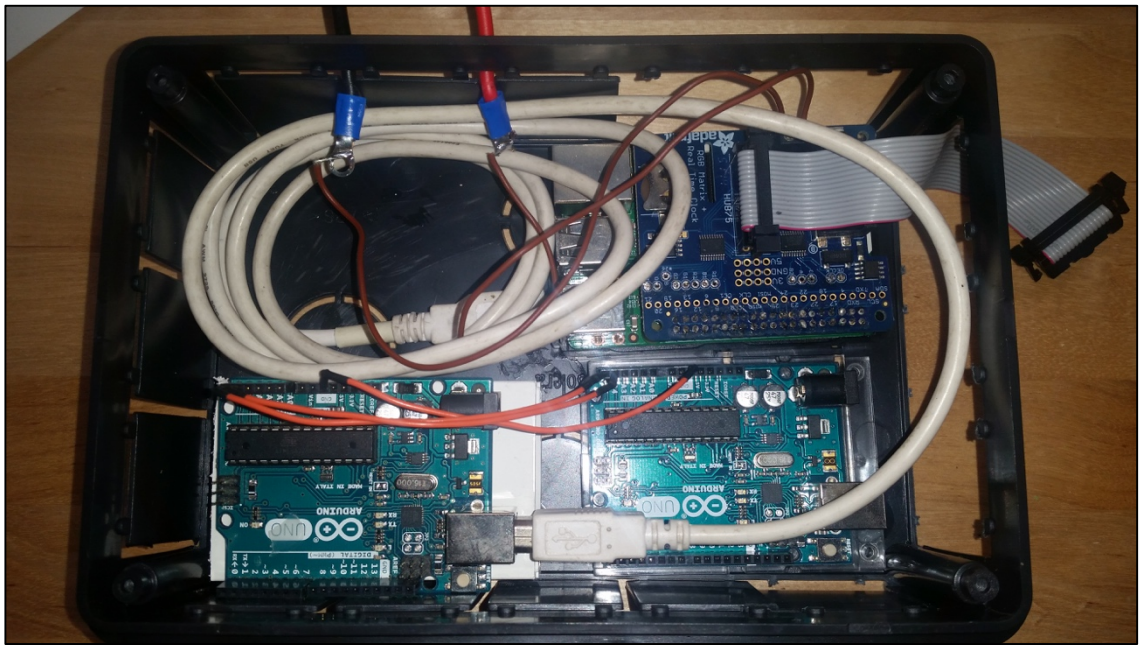


Figura 25. Interior del prototip

i exteriorment (figura 26):

Figura 26. Exterior del prototip

5. CONCLUSIONS

L'objectiu d'aquest projecte era programar una placa Raspberry per el control d'efectes visuals al ritme de la música i amb programació de codi lliure com son Processing, Ardunio i Python.

Això d'aconseguit amb uns punts molt marcats i la superació dels problemes que han anat apareixen durant el transcurs del projecte.

- S'ha estudiat, construït i programat un sistema de comunicació fiable, robust i fàcil d'entendre per comunicar el procés d'àudio en temps real amb l'execució dels programes en el mòdul led.

- S'ha estudiat i implementat el sistema de pins que es necessari perquè la pantalla de leds funcioni.

- S'ha estudiat i programat un llenguatge de codi lliure per dibuixar figures.

- S'ha estudiat diferents llenguatges de programació nous , que durant la carrera no s'han vist o quasi no san experimentat amb ells.

A nivell d'interactivitat amb l'usuari s'han obtingut resultat molt satisfactoris, ja que aquest encara que no tingui coneixements d'enginyeria ni coneixements previs del muntatge i programació, pot fer servir el mòdul nomes amb un ordinador i dues presses de corrent.

Al haver-se realitza tot el projecte amb components hardware destinats al aprenentatge i amb codi lliure, si hi ha una empresa amb necessitats d'incorporar aquesta tecnologia al seu projecte el cost serà molt mes baix que qualsevol altre empresa.

A nivell personal, gràcies a aquest projecte he posat en practica una gran quantitat de coneixements previs. Crec que he implementat una part de les tres branques principals de la meva titulació.

- He implementat una part hardware aplicant coneixements d'electrònica.
- He programat un conjunt d'instruccions ordenades per la visualització de figures, comunicació entre hardware, etc.
- He processat algun tipus de senyal en temps real.

He après a programar amb Python, que es un llenguatge molt important en el món empresarial i que per desgràcia no havia vist encara. També he aprofundit en el coneixement dels microcontroladors com Arduino i dels sistemes operatius Linux amb diferents plataformes(Raspberry).

També he adquirit un gran coneixement de com estructurar i desenvolupar un projecte d'enginyeria, estudiant pas a pas cada tecnologia i trobant la millor solució per cada problema. També he après a redactar de forma correcta i ordenada la memòria del mateix.

A destacar també que , gracies al desenvolupament d'aquest projecte i el continuo aprenentatge i problemes, he descobert i contactat amb una gran comunitat de fòrums relacionats amb petits projectes d'enginyeria. També m'ha despertat un gran interès per les empreses relacionades amb el disseny, programació e instal·lació de pantalles per mòduls led

5.1 Futures Aplicacions

Aquest projecte va néixer amb un futur molt marcat: El desenvolupament i la implementació de un gran sistema de mòduls Leds a la sala d'espectacles i discoteca Teatre Sant Cugat.

La meva trajectòria professional com a DJ de mes de 10 anys i com a encarregat del desenvolupament artístic de la sala em va fer decantar-me per aquest projecte. La idea es que amb una aportació econòmica important dissenyar una pantalla de leds de dimensions grans i controlar-la amb el prototip presentat. El projecte ja esta aprovat i amb un pressupost inicial de 1.500€. Agraeixo des d'aquest punt la confiança del equip de *JustForFun* que m'ha ajudat des de el primer moment per aquest projecte.

Un punt de futur per aquest projecte a gran escala és la anàlisis espectral de l'àudio per l'aprofitament d'aquest per el millor control dels efectes.

També queda una porta oberta a la innovació e investigació de tot allò relacionat amb el control de la Raspberry per efectes visuals com ara el control del Bus DMX512 de control de llums d'espectacle.

6. BIBLIOGRAFIA

Arduino.cc/es (2014, 10\02\2014). *Que és Arduino?* [Pàgina Web] Disponible: http://arduino.cc/es/Guide/Introduction

Arduino.cc/es (2014, 10\02\2014). *Master/Slave* [Pàgina Web] Disponible: https://www.arduino.cc/en/Tutorial/MasterWriter

Rapsberry Pi. *Que és raspberry?* [Pàgina Web]. Disponible: https://www.raspberrypi.org

Rapsberry Pi. *Instal·lació* [Pàgina Web]. Disponible: https://www.raspberrypi.org/downloads/

Bus I2C. *Que és i com funciona.* [Pàgina web] Disponible: https://es.wikipedia.org/wiki/I%C2%B2C

Component Adafruit .[Pàgina web] Disponible: https://www.adafruit.com/

Python. *Que és i com s'utilitza*. [Pàgina web] Disponible : https://www.python.org/

Repositori Python. *Matrix adafruit llibreries*.[Pàgina web] Disponible: https://github.com/hzeller

Processing. *Com és i com funciona*. [Pàgina web] Disponible: https://es.wikipedia.org/wiki/Processing

Processing. *Llibreria sound* [Pàgina web] Disponible: https://processing.org/reference/libraries/sound/index.html

7. ANNEXOS

Annex I Python:

```
from PIL import Image, ImageFont, ImageDraw
import Image
import time
from rgbmatrix import Adafruit_RGBmatrix
import sys
import random
import serial
import threading

arduino = serial.Serial('/dev/ttyACM0', baudrate=9600,
timeout=0.250)
x=1
posX=4
pos=4
# Rows and chain length are both required parameters:
matrix = Adafruit_RGBmatrix(16, 1)
# Bitmap example w/graphics prims
image = Image.new('RGB', (32, 32))# Can be larger than
matrix if wanted!!
image2 = Image.new('RGB', (32, 32))# Can be larger than
matrix if wanted!!
image21 = Image.new('RGB', (32, 32))
image22 = Image.new('RGB', (32, 32))
image23 = Image.new('RGB', (32, 32))
image24 = Image.new('RGB', (32, 32))
image25 = Image.new('RGB', (32, 32))
image26 = Image.new('RGB', (32, 32))
image27 = Image.new('RGB', (32, 32))
image3 = Image.new('RGB', (32, 32))
image4 = Image.new('RGB', (32, 32))
draw  = ImageDraw.Draw(image)    # Declare Draw instance
before prims
```

```
draw2  = ImageDraw.Draw(image2)    # Declare Draw instance
before prims
draw21  = ImageDraw.Draw(image21)
draw22  = ImageDraw.Draw(image22)
draw23  = ImageDraw.Draw(image23)
draw24  = ImageDraw.Draw(image24)
draw25  = ImageDraw.Draw(image25)
draw26  = ImageDraw.Draw(image26)
draw27  = ImageDraw.Draw(image27)
draw3  = ImageDraw.Draw(image3)
draw4  = ImageDraw.Draw(image4)
#draw.line((0, 0, 32, 0), fill="#ff0000", width = 3)
draw2.rectangle((0, 0, 4, 8), fill="#ADFF2F",
outline="#ADFF2F")
#draw3.rectangle((0, 0, 4, 8), fill="#FFFFFF",
outline="#FFFFFF")
draw4.rectangle((0, 0, 4, 16), fill="#00ffff",
outline="#00ffff")

def func2():
            matrix.SetImage(image2.im.id, 0, 4)
            draw2  = ImageDraw.Draw(image2)    # Declare
Draw instance before prims
            draw21  = ImageDraw.Draw(image21)
            draw22  = ImageDraw.Draw(image22)
            draw23  = ImageDraw.Draw(image23)
            draw24  = ImageDraw.Draw(image24)
            draw25  = ImageDraw.Draw(image25)
            draw26  = ImageDraw.Draw(image26)
            draw27  = ImageDraw.Draw(image27)
            if f3 == 'S':
                    matrix.Clear()
                    matrix.SetImage(image2.im.id, 0, 4)
            elif f3 == 'T':
                    matrix.Clear()
                    draw21.rectangle((0, 0, 4, 8),
fill="#00ff00", outline="#00ff00")
                    matrix.SetImage(image2.im.id, 0, 4)
                    matrix.SetImage(image21.im.id, 4, 4)

            elif f3 == 'U':
                    matrix.Clear()
                    draw21.rectangle((0, 0, 4, 8),
fill="#00ff00", outline="#00ff00")
                    draw22.rectangle((0, 0, 4, 8),
fill="#00ff00", outline="#00ff00")
                    matrix.SetImage(image2.im.id, 0, 4)
                    matrix.SetImage(image21.im.id, 4, 4)
                    matrix.SetImage(image22.im.id, 8, 4)
```

```
            elif f3 == 'V':
                    matrix.Clear()
                    draw21.rectangle((0, 0, 4, 8),
fill="#00ff00", outline="#00ff00")
                    draw22.rectangle((0, 0, 4, 8),
fill="#00ff00", outline="#00ff00")
                    draw23.rectangle((0, 0, 4, 8),
fill="#00ff00", outline="#00ff00")
                    matrix.SetImage(image2.im.id, 0, 4)
                    matrix.SetImage(image21.im.id, 4, 4)
                    matrix.SetImage(image22.im.id, 8, 4)
                    matrix.SetImage(image23.im.id, 12, 4)

            elif f3 == 'W':
                    matrix.Clear()
                    draw21.rectangle((0, 0, 4, 8),
fill="#00ff00", outline="#00ff00")
                    draw22.rectangle((0, 0, 4, 8),
fill="#00ff00", outline="#00ff00")
                    draw23.rectangle((0, 0, 4, 8),
fill="#00ff00", outline="#00ff00")
                    draw24.rectangle((0, 0, 4, 8),
fill="#00ff00", outline="#00ff00")
                    matrix.SetImage(image2.im.id, 0, 4)
                    matrix.SetImage(image21.im.id, 4, 4)
                    matrix.SetImage(image22.im.id, 8, 4)
                    matrix.SetImage(image23.im.id, 12, 4)
                    matrix.SetImage(image24.im.id, 16, 4)

            elif f3 == 'X':
                    matrix.Clear()
                    draw21.rectangle((0, 0, 4, 8),
fill="#00ff00", outline="#00ff00")
                    draw22.rectangle((0, 0, 4, 8),
fill="#00ff00", outline="#00ff00")
                    draw23.rectangle((0, 0, 4, 8),
fill="#00ff00", outline="#00ff00")
                    draw24.rectangle((0, 0, 4, 8),
fill="#00ff00", outline="#00ff00")
                    draw25.rectangle((0, 0, 4, 8),
fill="#f0ff00", outline="#f0ff00")
                    matrix.SetImage(image2.im.id, 0, 4)
                    matrix.SetImage(image21.im.id, 4, 4)
                    matrix.SetImage(image22.im.id, 8, 4)
                    matrix.SetImage(image23.im.id, 12, 4)
                    matrix.SetImage(image24.im.id, 16, 4)
                    matrix.SetImage(image25.im.id, 20, 4)
```

```
            elif f3 == 'Y':
                    matrix.Clear()
                    draw21.rectangle((0, 0, 4, 8),
fill="#ff0000", outline="#ff0000")
                    draw22.rectangle((0, 0, 4, 8),
fill="#ff0000", outline="#ff0000")
                    draw23.rectangle((0, 0, 4, 8),
fill="#ff0000", outline="#ff0000")
                    draw24.rectangle((0, 0, 4, 8),
fill="#ff0000", outline="#ff0000")
                    draw25.rectangle((0, 0, 4, 8),
fill="#f0ff00", outline="#f0ff00")
                    draw26.rectangle((0, 0, 4, 8),
fill="#f0ff00", outline="#f0ff00")
                    matrix.SetImage(image2.im.id, 0, 4)
                    matrix.SetImage(image21.im.id, 4, 4)
                    matrix.SetImage(image22.im.id, 8, 4)
                    matrix.SetImage(image23.im.id, 12, 4)
                    matrix.SetImage(image24.im.id, 16, 4)
                    matrix.SetImage(image25.im.id, 20, 4)
                    matrix.SetImage(image26.im.id, 24, 4)

            elif f3 == 'z':
                    matrix.Clear()
                    draw21.rectangle((0, 0, 4, 8),
fill="#ff0000", outline="#ff0000")
                    draw22.rectangle((0, 0, 4, 8),
fill="#ff0000", outline="#ff0000")
                    draw23.rectangle((0, 0, 4, 8),
fill="#ff0000", outline="#ff0000")
                    draw24.rectangle((0, 0, 4, 8),
fill="#ff0000", outline="#ff0000")
                    draw25.rectangle((0, 0, 4, 8),
fill="#f0ff00", outline="#f0ff00")
                    draw26.rectangle((0, 0, 4, 8),
fill="#f0ff00", outline="#f0ff00")
                    draw27.rectangle((0, 0, 4, 8),
fill="#ff0000", outline="#ff0000")
                    matrix.SetImage(image2.im.id, 0, 4)
                    matrix.SetImage(image21.im.id, 4, 4)
                    matrix.SetImage(image22.im.id, 8, 4)
                    matrix.SetImage(image23.im.id, 12, 4)
                    matrix.SetImage(image24.im.id, 16, 4)
                    matrix.SetImage(image25.im.id, 20, 4)
                    matrix.SetImage(image26.im.id, 24, 4)
                    matrix.SetImage(image27.im.id, 28, 4)
            del draw2
            del draw21
            del draw22
```

```
            del draw23
            del draw24
            del draw25
            del draw26
            del draw27



def func3():
        if f4 =='A':
            draw4.rectangle((0, 0, 4, 16), fill="#00ffff",
outline="#00ffff")
        elif f4 == 'B':
            draw4.rectangle((0, 0, 4, 16), fill="#fd00ff",
outline="#fd00ff")
        elif f4 == 'C':
            draw4.rectangle((0, 0, 4, 16), fill="#5af60a",
outline="#5af60a")
        elif f4 == 'D':
            draw4.rectangle((0, 0, 4, 16), fill="#0000ff",
outline="#0000ff")
        elif f4 == 'E':
            draw4.rectangle((0, 0, 4, 16), fill="#f0ff00",
outline="#f0ff00")
        posX= random.randrange(0,28,4)
        matrix.SetImage(image4.im.id, posX, 0)


def func4():
        if f4 =='A':
            draw3.rectangle((0, 0, 16, 16), fill="#fd00ff")
            draw3.rectangle((15, 0, 32, 16),
fill="#fd00ff")
            if f3 == 'S':

                matrix.Clear()
                matrix.SetImage(image3.im.id,0, 15)
            elif f3=='T':

                matrix.Clear()
                matrix.SetImage(image3.im.id,0, 13)
            elif f3=='U':

                matrix.Clear()
                matrix.SetImage(image3.im.id,0, 11)
            elif f3=='V':

                matrix.Clear()
                matrix.SetImage(image3.im.id,0, 9)
```

```
            elif f3=='W':

                matrix.Clear()
                matrix.SetImage(image3.im.id,0, 7)
            elif f3=='X':

                matrix.Clear()
                matrix.SetImage(image3.im.id,0, 5)
            elif f3=='Y':

                matrix.Clear()
                matrix.SetImage(image3.im.id,0, 3)
            elif f3=='Z':

                matrix.Clear()
                matrix.SetImage(image3.im.id,0, 1)


        elif f4 == 'B':
            draw3.rectangle((0, 0, 16, 16), fill="#00ffff")
            draw3.rectangle((15, 0, 32, 16),
fill="#00ffff")
            if f3 == 'S':

                matrix.Clear()
                matrix.SetImage(image3.im.id,0, 15)
            elif f3=='T':

                matrix.Clear()
                matrix.SetImage(image3.im.id,0, 13)
            elif f3=='U':

                matrix.Clear()
                matrix.SetImage(image3.im.id,0, 11)
            elif f3=='V':

                matrix.Clear()
                matrix.SetImage(image3.im.id,0, 9)
            elif f3=='W':

                matrix.Clear()
                matrix.SetImage(image3.im.id,0, 7)
            elif f3=='X':

                matrix.Clear()
                matrix.SetImage(image3.im.id,0, 5)
            elif f3=='Y':

                matrix.Clear()
```

```
                matrix.SetImage(image3.im.id,0, 3)
            elif f3=='Z':

                matrix.Clear()
                matrix.SetImage(image3.im.id,0, 1)


        elif f4 == 'C':
            draw3.rectangle((0, 0, 16, 16), fill="#0000ff")
            draw3.rectangle((15, 0, 32, 16),
fill="#0000ff")
            if f3 == 'S':

                matrix.Clear()
                matrix.SetImage(image3.im.id,0, 15)
            elif f3=='T':

                matrix.Clear()
                matrix.SetImage(image3.im.id,0, 13)
            elif f3=='U':

                matrix.Clear()
                matrix.SetImage(image3.im.id,0, 11)
            elif f3=='V':

                matrix.Clear()
                matrix.SetImage(image3.im.id,0, 9)
            elif f3=='W':

                matrix.Clear()
                matrix.SetImage(image3.im.id,0, 7)
            elif f3=='X':

                matrix.Clear()
                matrix.SetImage(image3.im.id,0, 5)
            elif f3=='Y':

                matrix.Clear()
                matrix.SetImage(image3.im.id,0, 3)
            elif f3=='Z':

                matrix.Clear()
                matrix.SetImage(image3.im.id,0, 1)
        elif f4 == 'D':
            draw3.rectangle((0, 0, 16, 16), fill="#00ff00")
            draw3.rectangle((15, 0, 32, 16),
fill="#00ff00")
            if f3 == 'S':
```

```
                matrix.Clear()
                matrix.SetImage(image3.im.id,0, 15)
            elif f3=='T':

                matrix.Clear()
                matrix.SetImage(image3.im.id,0, 13)
            elif f3=='U':

                matrix.Clear()
                matrix.SetImage(image3.im.id,0, 11)
            elif f3=='V':

                matrix.Clear()
                matrix.SetImage(image3.im.id,0, 9)
            elif f3=='W':

                matrix.Clear()
                matrix.SetImage(image3.im.id,0, 7)
            elif f3=='X':

                matrix.Clear()
                matrix.SetImage(image3.im.id,0, 5)
            elif f3=='Y':

                matrix.Clear()
                matrix.SetImage(image3.im.id,0, 3)
            elif f3=='Z':

                matrix.Clear()
                matrix.SetImage(image3.im.id,0, 1)
        elif f4 == 'E':
            draw3.rectangle((0, 0, 16, 16), fill="#f0ff00")
            draw3.rectangle((15, 0, 32, 16),
fill="#f0ff00")
            if f3 == 'S':

                matrix.Clear()
                matrix.SetImage(image3.im.id,0, 15)
            elif f3=='T':

                matrix.Clear()
                matrix.SetImage(image3.im.id,0, 13)
            elif f3=='U':

                matrix.Clear()
                matrix.SetImage(image3.im.id,0, 11)
            elif f3=='V':

                matrix.Clear()
```

```
                matrix.SetImage(image3.im.id,0, 9)
            elif f3=='W':

                matrix.Clear()
                matrix.SetImage(image3.im.id,0, 7)
            elif f3=='X':

                matrix.Clear()
                matrix.SetImage(image3.im.id,0, 5)
            elif f3=='Y':

                matrix.Clear()
                matrix.SetImage(image3.im.id,0, 3)
            elif f3=='Z':

                matrix.Clear()
                matrix.SetImage(image3.im.id,0, 1)

def worker():
        matrix.Clear()
        if f4 =='A':

            draw.arc((0,0, 4,4),start=0,end=360,
fill="#00ffff" )
        elif f4 == 'B':

            draw.arc((0,0, 4,4),start=0,end=360,
fill="#fd00ff" )
        elif f4 == 'C':

            draw.arc((0,0, 4,4),start=0,end=360,
fill="#5af60a" )
        elif f4 == 'D':

            draw.arc((0,0, 4,4),start=0,end=360,
fill="#0000ff" )
        elif f4 == 'E':

            draw.arc((0,0, 4,4),start=0,end=360,
fill="#f0ff00" )

        posX= random.randrange(0,28)
        posY= random.randrange(0,12)
        i = 0
        for i in range (20):
            matrix.SetImage(image.im.id, posX, 16-i)
            time.sleep(0.025)
```

```
def func5():
        t = threading.Thread(target=worker)
        t.start()


def func1():
        line = arduino.readline()
        #tecla = sys.stdin.read(1)
        #print(line)
        f = line[0:1]
        f2 = line[1:2]
        f3 = line[2:3]
        f4 = line[3:4]
        draw  = ImageDraw.Draw(image)
        if f == "0" and f2 == "F":
            #print(f)
            #func4()
            #draw3.rectangle((0, 0, 16, 16),
fill="#00ffff")
            #draw3.rectangle((15, 0, 32, 16),
fill="#00ffff")
            func4()
        if f == "0" and f2 == "E":
            #print(f3)
            #draw.rectangle((0, 0, 8, 8), fill=0,
outline="#00FF00")
            func5()

            #matrix.SetImage(image.im.id, 27, 1)


        if  f2 == "D":
            #print(f)
            func3()
        if f == "0" and f2 == "G":
            #print(f3)
           func2()
        if f == "C" :
            print(f)
            del draw
            matrix.Clear()




while x>0:
        line = arduino.readline()
```

```
#tecla = sys.stdin.read(1)
#print(line)
f = line[0:1]
f2 = line[1:2]
f3 = line[2:3]
f4 = line[3:4]
func1()
print(line)
```

Annex II Arduino:

```
#include <Wire.h>

int val;
 int ledPin = 10;
 byte x = 0;
 int in[3];
 char buf[2]="0B";
 void setup() {
     Wire.begin();
    pinMode(ledPin, OUTPUT);
    Serial.begin(9600);
    pinMode(13, OUTPUT);
 }

 void loop() {

      while (Serial.available()) {
       int val = Serial.read();
       in[0] = val;
       int val2 = Serial.read();
       in[1] = val2;
       int val3 = Serial.read();
       in[2] = val3;
       int val4 = Serial.read();
       in[3] = val4;
            if(val2==65 and val==49 and val3==83 and
val4==65 ){
              digitalWrite(13, HIGH);
              Wire.beginTransmission(8);
              Wire.write("0FSA");
              Wire.endTransmission();
            }else if(val2==65 and val==49 and val3==84 and
val4==65 ){
              digitalWrite(13, HIGH);
              Wire.beginTransmission(8);
              Wire.write("0FTA");
              Wire.endTransmission();
            }else if(val2==65 and val==49 and val3==85 and
val4==65 ){
              digitalWrite(13, HIGH);
              Wire.beginTransmission(8);
              Wire.write("0FUA");
              Wire.endTransmission();
            }else if(val2==65 and val==49 and val3==86 and
val4==65 ){
```

```
              digitalWrite(13, HIGH);
              Wire.beginTransmission(8);
              Wire.write("0FVA");
              Wire.endTransmission();
            }else if(val2==65 and val==49 and val3==87 and
val4==65 ){
              digitalWrite(13, HIGH);
              Wire.beginTransmission(8);
              Wire.write("0FWA");
              Wire.endTransmission();
            }else if(val2==65 and val==49 and val3==88 and
val4==65 ){
              digitalWrite(13, HIGH);
              Wire.beginTransmission(8);
              Wire.write("0FXA");
              Wire.endTransmission();
            }else if(val2==65 and val==49 and val3==89 and
val4==65 ){
              digitalWrite(13, HIGH);
              Wire.beginTransmission(8);
              Wire.write("0FYA");
              Wire.endTransmission();
            }else if(val2==65 and val==49 and val3==90 and
val4==65 ){
              digitalWrite(13, HIGH);
              Wire.beginTransmission(8);
              Wire.write("0FZA");
              Wire.endTransmission();
            }else if (val2 == 65 and val==49 and val3==83
and val4==66){
              digitalWrite(13, LOW);
              Wire.beginTransmission(8);
              Wire.write("0FSB");
              Wire.endTransmission();
            }else if (val2 == 65 and val==49 and val3==84
and val4==66){
              digitalWrite(13, LOW);
              Wire.beginTransmission(8);
              Wire.write("0FTB");
              Wire.endTransmission();
            }else if (val2 == 65 and val==49 and val3==85
and val4==66){
              digitalWrite(13, LOW);
              Wire.beginTransmission(8);
              Wire.write("0FUB");
              Wire.endTransmission();
            }else if (val2 == 65 and val==49 and val3==86
and val4==66){
              digitalWrite(13, LOW);
```

```
              Wire.beginTransmission(8);
              Wire.write("0FVB");
              Wire.endTransmission();
            }else if (val2 == 65 and val==49 and val3==87
and val4==66){
              digitalWrite(13, LOW);
              Wire.beginTransmission(8);
              Wire.write("0FWB");
              Wire.endTransmission();
            }else if (val2 == 65 and val==49 and val3==88
and val4==66){
              digitalWrite(13, LOW);
              Wire.beginTransmission(8);
              Wire.write("0FXB");
              Wire.endTransmission();
            }else if (val2 == 65 and val==49 and val3==89
and val4==66){
              digitalWrite(13, LOW);
              Wire.beginTransmission(8);
              Wire.write("0FYB");
              Wire.endTransmission();
            }else if (val2 == 65 and val==49 and val3==90
and val4==66){
              digitalWrite(13, LOW);
              Wire.beginTransmission(8);
              Wire.write("0FZB");
              Wire.endTransmission();
            }else if (val2 == 65 and val==49 and val3==83
and val4==67){
              digitalWrite(13, LOW);
              Wire.beginTransmission(8);
              Wire.write("0FSC");
              Wire.endTransmission();
            }else if (val2 == 65 and val==49 and val3==84
and val4==67){
              digitalWrite(13, LOW);
              Wire.beginTransmission(8);
              Wire.write("0FTC");
              Wire.endTransmission();
            }else if (val2 == 65 and val==49 and val3==85
and val4==67){
              digitalWrite(13, LOW);
              Wire.beginTransmission(8);
              Wire.write("0FUC");
              Wire.endTransmission();
            }else if (val2 == 65 and val==49 and val3==86
and val4==67){
              digitalWrite(13, LOW);
              Wire.beginTransmission(8);
```

```
              Wire.write("0FVC");
              Wire.endTransmission();
            }else if (val2 == 65 and val==49 and val3==87
and val4==67){
              digitalWrite(13, LOW);
              Wire.beginTransmission(8);
              Wire.write("0FWC");
              Wire.endTransmission();
            }else if (val2 == 65 and val==49 and val3==88
and val4==67){
              digitalWrite(13, LOW);
              Wire.beginTransmission(8);
              Wire.write("0FXC");
              Wire.endTransmission();
            }else if (val2 == 65 and val==49 and val3==89
and val4==67){
              digitalWrite(13, LOW);
              Wire.beginTransmission(8);
              Wire.write("0FYC");
              Wire.endTransmission();
            }else if (val2 == 65 and val==49 and val3==90
and val4==67){
              digitalWrite(13, LOW);
              Wire.beginTransmission(8);
              Wire.write("0FZC");
              Wire.endTransmission();
            }else if (val2 == 65 and val==49 and val3==83
and val4==68){
              digitalWrite(13, LOW);
              Wire.beginTransmission(8);
              Wire.write("0FSD");
              Wire.endTransmission();
            }else if (val2 == 65 and val==49 and val3==84
and val4==68){
              digitalWrite(13, LOW);
              Wire.beginTransmission(8);
              Wire.write("0FTD");
              Wire.endTransmission();
            }else if (val2 == 65 and val==49 and val3==85
and val4==68){
              digitalWrite(13, LOW);
              Wire.beginTransmission(8);
              Wire.write("0FUD");
              Wire.endTransmission();
            }else if (val2 == 65 and val==49 and val3==86
and val4==68){
              digitalWrite(13, LOW);
              Wire.beginTransmission(8);
              Wire.write("0FVD");
```

```
              Wire.endTransmission();
            }else if (val2 == 65 and val==49 and val3==87
and val4==68){
              digitalWrite(13, LOW);
              Wire.beginTransmission(8);
              Wire.write("0FWD");
              Wire.endTransmission();
            }else if (val2 == 65 and val==49 and val3==88
and val4==68){
              digitalWrite(13, LOW);
              Wire.beginTransmission(8);
              Wire.write("0FXD");
              Wire.endTransmission();
            }else if (val2 == 65 and val==49 and val3==89
and val4==68){
              digitalWrite(13, LOW);
              Wire.beginTransmission(8);
              Wire.write("0FYD");
              Wire.endTransmission();
            }else if (val2 == 65 and val==49 and val3==90
and val4==68){
              digitalWrite(13, LOW);
              Wire.beginTransmission(8);
              Wire.write("0FZD");
              Wire.endTransmission();
            }else if (val2 == 65 and val==49 and val3==83
and val4==69){
              digitalWrite(13, LOW);
              Wire.beginTransmission(8);
              Wire.write("0FSE");
              Wire.endTransmission();
            }else if (val2 == 65 and val==49 and val3==84
and val4==69){
              digitalWrite(13, LOW);
              Wire.beginTransmission(8);
              Wire.write("0FTE");
              Wire.endTransmission();
            }else if (val2 == 65 and val==49 and val3==85
and val4==69){
              digitalWrite(13, LOW);
              Wire.beginTransmission(8);
              Wire.write("0FUE");
              Wire.endTransmission();
            }else if (val2 == 65 and val==49 and val3==86
and val4==69){
              digitalWrite(13, LOW);
              Wire.beginTransmission(8);
              Wire.write("0FVE");
              Wire.endTransmission();
```

```
            }else if (val2 == 65 and val==49 and val3==87
and val4==69){
              digitalWrite(13, LOW);
              Wire.beginTransmission(8);
              Wire.write("0FWE");
              Wire.endTransmission();
            }else if (val2 == 65 and val==49 and val3==88
and val4==69){
              digitalWrite(13, LOW);
              Wire.beginTransmission(8);
              Wire.write("0FXE");
              Wire.endTransmission();
            }else if (val2 == 65 and val==49 and val3==89
and val4==69){
              digitalWrite(13, LOW);
              Wire.beginTransmission(8);
              Wire.write("0FYE");
              Wire.endTransmission();
            }else if (val2 == 65 and val==49 and val3==90
and val4==69){
              digitalWrite(13, LOW);
              Wire.beginTransmission(8);
              Wire.write("0FZ32E");
              Wire.endTransmission();
            }else if (val2 == 66 and val==49){
              digitalWrite(13, LOW);
              Wire.beginTransmission(8);
              Wire.write("C0FF");
              Wire.endTransmission();
            }else if (val2==65 and val==50 and val4==65){
              digitalWrite(13, LOW);
              Wire.beginTransmission(8);
              Wire.write("0EFA");
              Wire.endTransmission();
            }else if (val2==65 and val==50 and val4==66){
              digitalWrite(13, LOW);
              Wire.beginTransmission(8);
              Wire.write("0EFB");
              Wire.endTransmission();
            }else if (val2==65 and val==50 and val4==67){
              digitalWrite(13, LOW);
              Wire.beginTransmission(8);
              Wire.write("0EFC");
              Wire.endTransmission();
            }else if (val2==65 and val==50 and val4==68){
              digitalWrite(13, LOW);
              Wire.beginTransmission(8);
              Wire.write("0EFD");
              Wire.endTransmission();
```

```
}else if (val2==65 and val==50 and val4==69){
  digitalWrite(13, LOW);
  Wire.beginTransmission(8);
  Wire.write("0EFE");
  Wire.endTransmission();
}else if (val2==66 and val==50){
  digitalWrite(13, LOW);
  Wire.beginTransmission(8);
  Wire.write("C0FF");
  Wire.endTransmission();
}else if (val2==65 and val==51 and val4==65){
  digitalWrite(13, LOW);
  Wire.beginTransmission(8);
  Wire.write("0DFA");
  Wire.endTransmission();
}else if (val2==65 and val==51 and val4==66){
  digitalWrite(13, LOW);
  Wire.beginTransmission(8);
  Wire.write("0DFB");
  Wire.endTransmission();
}else if (val2==65 and val==51 and val4==67){
  digitalWrite(13, LOW);
  Wire.beginTransmission(8);
  Wire.write("0DFC");
  Wire.endTransmission();
}else if (val2==65 and val==51 and val4==68){
  digitalWrite(13, LOW);
  Wire.beginTransmission(8);
  Wire.write("0DFD");
  Wire.endTransmission();
}else if (val2==65 and val==51 and val4==69){
  digitalWrite(13, LOW);
  Wire.beginTransmission(8);
  Wire.write("0DFE");
  Wire.endTransmission();
} else if (val2==66 and val==51){
  digitalWrite(13, LOW);
  Wire.beginTransmission(8);
  Wire.write("C0FF");
  Wire.endTransmission();
}else if (val2==65 and val==52 and val3==83){
  digitalWrite(13, LOW);
  Wire.beginTransmission(8);
  Wire.write("0GSE");
  Wire.endTransmission();
}else if (val2==65 and val==52 and val3==84){
  digitalWrite(13, LOW);
  Wire.beginTransmission(8);
  Wire.write("0GTE");
```

```
            Wire.endTransmission();
          }else if (val2==65 and val==52 and val3==85){
            digitalWrite(13, LOW);
            Wire.beginTransmission(8);
            Wire.write("0GUE");
            Wire.endTransmission();
          }else if (val2==65 and val==52 and val3==86){
            digitalWrite(13, LOW);
            Wire.beginTransmission(8);
            Wire.write("0GVE");
            Wire.endTransmission();
          }else if (val2==65 and val==52 and val3==87){
            digitalWrite(13, LOW);
            Wire.beginTransmission(8);
            Wire.write("0GWE");
            Wire.endTransmission();
          }else if (val2==65 and val==52 and val3==88){
            digitalWrite(13, LOW);
            Wire.beginTransmission(8);
            Wire.write("0GXE");
            Wire.endTransmission();
          }else if (val2==65 and val==52 and val3==89){
            digitalWrite(13, LOW);
            Wire.beginTransmission(8);
            Wire.write("0GYE");
            Wire.endTransmission();
          }else if (val2==65 and val==52 and val3==90){
            digitalWrite(13, LOW);
            Wire.beginTransmission(8);
            Wire.write("0GZE");
            Wire.endTransmission();
          }else if (val2==66 and val==52){
            digitalWrite(13, LOW);
            Wire.beginTransmission(8);
            Wire.write("CGSF");
            Wire.endTransmission();
          }

      }

        delay(10);
}
```

Annex III Processing:

```
import processing.sound.*;
import processing.serial.*; //Importamos la librería Serial

Serial myPort; //Nombre del puerto serie
AudioIn input;
Amplitude rms;
char a;
char b;
char c;
char d;
int scale=1;
int x;

String str1 ;
char data[] = {'1', 'A', 'S','A'};


void setup() {
    size(640,360);
    background(255);
     String portName = Serial.list()[1];
    myPort = new Serial(this, portName, 9600);
    input = new AudioIn(this, 0);
    input.start();
    rms = new Amplitude(this);
    rms.input(input);
}
//funcio que retorna el valor de la tecla presionada
void miFuncion(){
      if (key == '1'){
          a='1';
        }else if (key == '2'){
          a='2';
        }else if (key == '3'){
          a='3';
        }else if (key == '4'){
          a='4';
        }else if (key == '5'){
          a='5';
        }
        if (key == 'q'){
          d='A';
        }else if (key == 'w'){
          d='B';
        }else if (key == 'e'){
```

```
          d='C';
        }else if (key == 'r'){
          d='D';
        }else if (key == 't'){
          d='E';
        }
}

void draw() {
    if (keyPressed == true) {
        miFuncion();
    }
    data[0]=a;
    data[3]=d;
    background(125,255,125);
    scale=int(map(rms.analyze(), 0, 0.5, 1, 350));
    noStroke();
    fill(255,0,150);
    //ellipse(width/2, height/2, 1*scale, 1*scale);
    text("Programas:   Programa 1   -   Programa 2   -
Programa 3   -   Programa 4", 14, 60);
    text("teclas:           -->1                -->2
-->3                -->4 ", 14, 80);
    text("Colors:   Cyan   -   Violeta   -   Blau   -
Verd   -   Groc", 14, 120);
    text("teclas:    Q             W             E
R             T", 14, 140);
    if (scale >20){
        b='A';
    } else b='B';
    data[1]=b;
    if (scale > 0 && scale<=10){
       c='S';
    }else if (scale > 10 && scale<=19){
      c='T';
    }else if (scale > 19 && scale<=27){
      c='U';
    }else if (scale > 27 && scale<=33){
      c='V';
    }else if (scale > 33 && scale<=38){
       c='W';
    }else if (scale > 38 && scale<=42){
      c='X';
    }else if (scale > 42 && scale<=45){
      c='Y';
    }else if (scale > 45 ){
      c='Z';
    }
    data[2]=c;
```

```
String str2 = new String(data);
myPort.write(str2);
println();
```